64'er
SONDERHEFT C 64-GRUNDWISSEN

SONDERHEFT 5/1986 Lit. 12000/hfl. 18.-/dkr. 68,- ÖS 100,-/Sfr. 14,- DM 14,-

Markt&Technik

64'er



Der C 64: ausführlich und verständlich für alle

Kurse zum Mitmachen
- ★ So funktioniert der C 64
- ★ Grafik
- ★ Sprites
- ★ Musik

Die ideale Ausstattung
- ★ Drucker
- ★ Monitore
- ★ Floppy, Datasette
- ★ Programme für jeden Zweck

Unentbehrliche Programme zum Abtippen

Computer-Lexikon Tips & Tricks

Alle Programme auch auf Diskette erhältlich



Bücher zum Commodore 64

Lust statt Frust

Es kribbelt Ihnen in den Fingern. Der Herzschlag hat sich um einige Takte erhöht, die Handflächen sind feucht und leicht nervös öffnen Sie die Verpackung Ihres neuen C 64. Lange genug haben Sie die Computerszene beobachtet. Die Entscheidung, sich einen C 64 zu kaufen, fiel Ihnen deshalb auch nicht besonders schwer. Kein Wunder, denn jeder spricht von ihm. Ihr Freund, Ihr Kollege oder, wenn Sie Schüler sind, garantiert eine Handvoll Mitschüler schwärmt schon lange von »seinem« Computer. Stolz erzählen sie, was sie schon alles gemacht haben. Sie sind begeistert von den Spielen, von den grafischen Möglichkeiten. Der Sound, die Musik, die dem C 64 entlockt werden kann, findet begeisterte Zuhörer.



Als frischgebackener Computerbesitzer lassen Sie sich sicherlich gerne von soviel Enthusiasmus anstecken. Ihre ersten Programmierversuche, ähnliche Ergebnisse auf Ihrem C 64 zu zaubern, dürften jedoch nicht so besonders überzeugend sein. Woran liegt's? Vielleicht sind Sie der Werbung auf den Leim gegangen, die behauptet, »computern« sei eine der einfachsten Sachen auf der Welt, alles laufe sozusagen von selbst. In Wirklichkeit ist die Arbeit mit einem Computer jedoch eine zeitraubende Angelegenheit. Erfolge müssen hart erkämpft werden. Dagegen sind wirkliche Mißerfolge so gut wie ausgeschlossen. Und das macht das Hobby programmieren so unwahrscheinlich interessant. Wenn Sie am Anfang noch nicht einmal wissen, wie man einen Text wie gewünscht auf den Bildschirm bringt, so ist diese Aufgabe nach wenigen Stunden schon längst Routine geworden und Sie stehen bereits vor neuen Problemen. Aber auch diese werden meistens gelöst. Und wenn nicht, dann gibt es ja noch so viel zu entdecken und zu versuchen. Dieses Spielchen setzt sich fort, auch wenn Sie schon längst Profi sind.

Nun brauchen Sie jedoch das Rad nicht noch einmal zu erfinden. Wir wissen aus eigener Erfahrung, daß es manchmal nur an der mangelnden Information liegt, wenn bestimmte Probleme nicht gelöst werden können. Aus diesem Grunde haben wir ein Sonderheft zusammengestellt, in dem gerade Sie als Einsteiger genau das Grundwissen erhalten, das Sie brauchen. Den Computer, also Ihren C 64, kennenzulernen, bedeutet nicht nur zu wissen, wie man programmiert, sondern es erleichtert Ihnen das Verständnis erheblich, wenn Sie auch wissen, wie er funktioniert. Deshalb sollten Sie mit uns eine Exkursion durch den C 64 machen. Wir erklären Ihnen, wie der Speicher, das Gedächtnis, aufgebaut ist und stellen Ihnen die wichtigsten Begriffe vor, mit denen Sie immer wieder zu tun haben werden, die andere benutzen, als hätten sie diese schon in der Wiege gelernt.

Vielleicht haben Sie sich schon gewundert, wozu die verschiedenen Anschlüsse des Commodore eingesetzt werden können. Bei dem seriellen Port dürfte es keine Probleme geben, er ist für den Anschluß eines Druckers oder des Diskettenlaufwerks notwendig. Auch der Joystick-Port und der Kassettenport lassen wenig Fragen aufkommen. Anders sieht es schon bei dem User- und dem Expansions-Port aus. Was mit ihnen gemacht werden kann, erfahren Sie im Artikel über die Ports des C 64.

Obwohl der vorhin erwähnte serielle Port selbst wenig Überraschungen bietet, so haben es die Geräte, die daran anzuschließen sind, in sich. Das Diskettenlaufwerk zum Speichern von Programmen und Daten, manchmal auch nur kurz »Floppy« genannt, ist neben dem Drucker wohl die interessanteste Erweiterung Ihres C 64. Wir erklären Ihnen, was es damit auf sich hat, zu welchen Leistungen die Floppy fähig ist und wie Daten gespeichert werden.

Sind Sie etwa der Meinung, daß Sie auch ohne Drucker auskommen? Dann stellen Sie sich diese Frage doch in einigen Wochen oder Monaten noch einmal. Garantiert haben Sie dann eine 180-Grad-Wendung vollzogen. Kann man auf das Diskettenlaufwerk zu Gunsten der Datasette (vorläufig) noch verzichten, so wird der Wunsch nach einem Drucker bald übermächtig. Damit Sie beim Kauf keine Fehlentscheidung treffen, sollten Sie auf unsere jahrelange Erfahrung zurückgreifen. Wir stellen Ihnen die verbreitetsten und für Sie besten Drucker vor, für jeden Geldbeutel und für jeden Zweck. Mit unseren Empfehlungen können Sie eigentlich nichts mehr falsch machen.

Programmieren ist ein Zauberwort. Es bedeutet Entspannung, Spaß, Erfolgserlebnisse, Spiel und Spannung. Doch aller Anfang ist schwer. Deshalb führen wir Sie ganz behutsam und ausführlich in die Basic-Programmierung ein. Ein Kurs zum Mitmachen, mit vielen Tips und Tricks. Auch wenn Ihnen noch viele Basic-Befehle unklar sind, hier bekommen Sie Durchblick und verlieren die Scheu vor Fehlern. Nicht daß Sie jetzt denken, Sie könnten anschließend fehlerfrei programmieren, das kann keiner. Sie werden es bei jedem Programm merken, der C 64 ist kleinlicher, als es ein Mensch je sein wird. Die Fehlermeldungen, die er dann auswirft, lassen auch oft zu wünschen übrig. Doch auch hier helfen wir Ihnen weiter. Im Artikel »Mein Computer versteht mich nicht« erklären wir jede Fehlermeldung des C 64, beschreiben die Ursache und wie man den Fehler findet und behebt. Selbst für fortgeschrittene Programmierer gibt es einige verblüffende Erkenntnisse.

Gehört die Tätigkeit Programmieren auch zu den angenehmsten Beschäftigungen eines C 64-Besitzers, so brauchen Sie dennoch einige wichtige Hilfsprogramme, die Ihnen bei der Arbeit helfen und sie erleichtern. Die wichtigsten haben wir für Sie zusammengestellt: ein Kopierprogramm für Datasette und Diskette, Hypra-Load und Turbotape, die die Arbeit mit dem Diskettenlaufwerk und der Datasette erheblich beschleunigen und ein Sprite-Editor, mit dem Sie sehr einfach Sprites erstellen können.

Versäumen Sie auf keinen Fall, sich die »Fragen & Antworten« durchzulesen. Auf acht Seiten haben wir die wichtigsten und am häufigsten gestellten Fragen beantwortet. Eine Fundgrube für jeden Programmierer und C 64-Besitzer. Das gilt übrigens auch für die »PEEK-, POKE- und SYS-Kiste« sowie, last not least, für die vielen »Tips & Tricks für Einsteiger«. Wir hoffen, Ihnen mit diesem Sonderheft eine wichtige und interessante Hilfe in die Hand zu geben.

(Georg Klinge)

PROGRAMM-SERVICE



Bestellungen in der Schweiz: Markt&Technik Vertriebs AG, Kollerstrasse 3, CH-6300 Zug, Tel. 042/41 56 56
Bestellungen in Österreich: Bücherzentrum Meidling, Schönbrunner Straße 261, A-1120 Wien, Tel. 0222/83 31 96,
Microcomput-ique E. Schiller, Fasangasse 21, A-1030 Wien, Tel. 0222/78 56 61,
Ueberreuter Media Handels- und Verlagsgesellschaft mbH, Alser Straße 24, A-1091 Wien, Tel. 0222/48 15 38-0
Bestellungen aus anderen Ländern bitte per Auslandspostanweisung!

Das Angebot dieser Ausgabe:

Wer keine Zeit oder keine Lust hat, alle Programme selbst in mühevoller Kleinarbeit abzuschreiben, kann wieder auf den bewährten Programm-Service zurückgreifen. Alle Programme, die mit dem Diskettensymbol ▪ im Inhaltsverzeichnis gekennzeichnet sind, gibt's auf Diskette.

1 Diskette
Bestell-Nr. L6 86 S5D — **DM 29,90*** (sFr. 24,90/öS 299,-*)

* inkl. MwSt. Unverbindliche Preisempfehlung

Spitzmon: Der Monitor zum Ascompiler Basic und Compilerversion — S. 69
Tips und Tricks für Profis — S. 77
Sound-Editor — S. 80
CIA: Echtzeituhr/DFÜ — S. 98
Schreiberling: Märchenstunde für Drucker MPS 801/802/803 — S. 102

Programme aus früheren Ausgaben:

64'er-Ausgabe 4/86
Bestell-Nr. L6 86 04D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Quizmaster – Prüfungsvorbereitungen oder Party-Gag — S. 53
Hypra Basic – Erstellen persönlicher Basicerweiterungen — S. 58
Druckroutine zu DATABASE (DB II) – Endlich Datensätze auf dem Drucker ausgeben — S. 63
Hardmaker – Grafik-Bilder aus fast allen Programmen drucken — S. 67
Synchro Justage – Jetzt ist Schluß mit »LOAD ERROR« — S. 77
Micro-Tagebuch – Niemand hat Zutritt zu Ihren privaten Aufzeichnungen — S. 77
Ex-Line – Basiczeilen mit 252 statt 80 Zeichen — S. 78
Soft-Flash – kleiner Trick an der Floppy — S. 79
Strich-Cursor – verleiht Ihrem Cursor ein äußerst professionelles Aussehen — S. 79
Upside Down – Dreht den Bildschirm um 180 Grad — S. 79
Disk-Optimizer – Optimale Ausnutzung Ihrer Disketten. Basic und Compilerversion — S. 80
Apfelmännchen – Diashow für Grafiken — S. 84
Autochange – Ihr Commodore 128 springt automatisch in den richtigen Modus — S. 85
Taktzyklen – Exaktes Ausmessen eines Unterprogrammes hilft, Laufzeiten zu verringern. Für Basic und Maschinenprogramme — S. 86

64'er-Ausgabe 3/86
Bestell-Nr. L6 86 03D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Eingabehilfe Checksummer V3 und MSE — S. 55
Kudiplo – Funktion diskutieren und plotten — S. 58
64'er DOS – alle Funktionen der 1541 beschleunigen — S. 65
Shapes auf dem C64 mit Demo-Programm — S. 71
Auto-Old: letzte Rettung nach »new« — S. 79
Englisch für Fortgeschrittene — S. 80
HiRes-Scrolling mit Demo-Programm und Quelltext — S. 81
1520-Plotter als Drucker — S. 82
Laufschriftgenerator – ruckfreie Laufschrift für eigene Programme — S. 83
Centronics-Interface mit Quelltext für den C128 — S. 84
View Picture – Endlich auch farbige Hi-Eddi-Bilder für eigene Programme — S. 91

64'er-Ausgabe 2/86
Bestell-Nr. L6 86 02D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

text-transposer — S. 51
Garbage Collection: Müllabfuhr für Strings in max. 1 Sekunde — S. 53
Eingabehilfe: MSE + Checksummer — S. 57
Profiauflösung für MPS 801/803 — S. 59
Software zum 64'er Eprom-Programmiergerät — S. 65

64'er-Ausgabe 1/86
Bestell-Nr. L6 86 01D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Checksummer V3 — S. 54
MSE V1.0 — S. 54
Datawork 1.1 — S. 56
Ascompiler — S. 60
Hardcopy — S. 67
Life — S. 69
Vergleich von Programmen — S. 77
MSE-Hex-Tastatur — S. 78
Die unmögliche Uhr — S. 78
Screenlarger + Demo — S. 81
C128 – Grafikprogramme:
– Fensterrose — S. 131
– Spiralen — S. 133
– Box-Befehl — S. 134
IEEE-Generator — S. 147

64'er-Ausgabe 12/85
Bestell-Nr. L6 85 12D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)
Bestell-Nr. L6 85 12K Kassette
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 11/85
Bestell-Nr. L6 85 11A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 10/85
Bestell-Nr. L6 85 10A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 9/85
Bestell-Nr. L6 85 09A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 8/85
Bestell-Nr. L6 85 08A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 7/85
Bestell-Nr. L6 85 07A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 6/85
Bestell-Nr. L6 85 06A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 5/85
Bestell-Nr. L6 85 05A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 4/85
Bestell-Nr. L6 85 04A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 3/85
Bestell-Nr. L6 85 03A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 2/85
Bestell-Nr. L6 85 02A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Ausgabe 1/85
Bestell-Nr. L6 85 01A
DM 29,90* (sFr. 24,90/öS 299,-*)

64'er-Sonderhefte

Sonderheft 4/86 – Abenteuer
Bestell-Nr. L6 86 S4D 2 Disketten
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 3/86 – C16, C116, VC20, Plus 4
1 Diskette für VC20 und C16/116:
Bestell-Nr. L6 86 S3 CD
DM 29,90* (sFr. 24,90/öS 299,-*)
1 Kassette für VC20:
Bestell-Nr. L6 86 S3 KV
DM 19,90* (sFr. 17,-/öS 199,-*)
1 Kassette für C16:
Bestell-Nr. L6 86 S3 KC
DM 19,90* (sFr. 17,-/öS 199,-*)

Sonderheft 2/86 – Tips&Tricks
Bestell-Nr. L6 86 S2D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Sonderheft 1/86 – C128er
Bestell-Nr. L6 86 S1D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)

Sonderheft 8/85 – Assembler
Bestell-Nr. L6 85 S8D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)
Bestell-Nr. L6 85 S8K Kassette
DM 19,90* (sFr. 17,-/öS 199,-*)

Sonderheft 7/85 – Professionelle Anwendungen
Bestell-Nr. L6 85 S7D 2 Disketten
DM 34,90* (sFr. 29,50/öS 349,-*)
Bestell-Nr. L6 85 S7K 4 Kassetten
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 6/85 – Top-Themen
Bestell-Nr. L6 85 S6 2 Disketten
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 5/85 – Floppy, Datasette
Bestell-Nr. L6 85 S5D Diskette
DM 29,90* (sFr. 24,90/öS 299,-*)
Bestell-Nr. L6 85 S5K Kassette
DM 19,90* (sFr. 17,-/öS 199,-*)

Sonderheft 4/85 – Grafik
Bestell-Nr. L6 85 S4A
DM 29,90* (sFr. 24,90/öS 299,-*)

Sonderheft 3/85 – Spiele
Bestell-Nr. L6 85 S3 A 2 Disketten
DM 34,90* (sFr. 29,50/öS 349,-*)

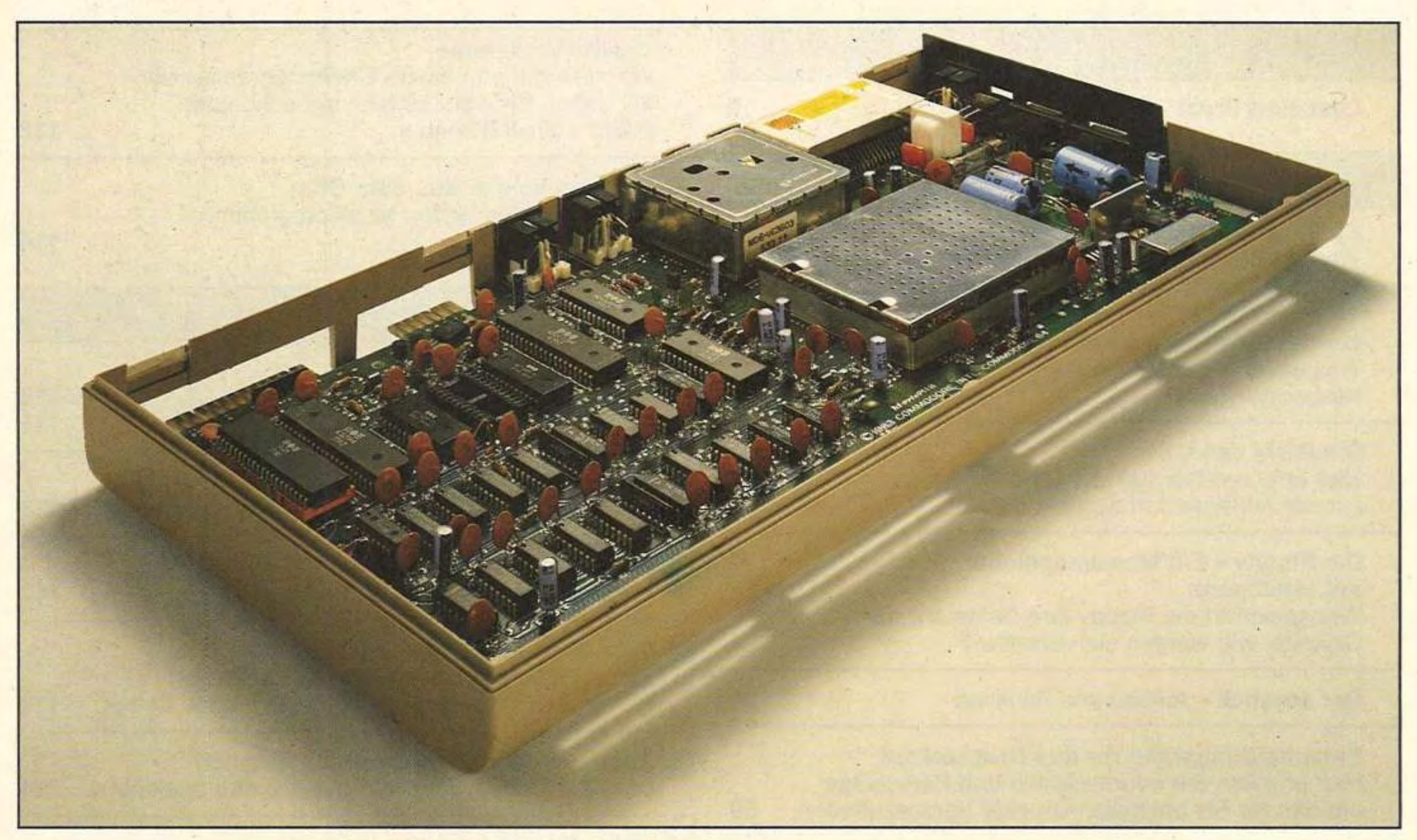
Sonderheft 2/85 – Abenteuerspiele
Bestell-Nr. L6 85 S2
DM 34,90* (sFr. 29,50/öS 349,-*)

Sonderheft 1/85 – Tips&Tricks (2. überarb. Auflage)
Bestell-Nr. CB 023 Floppy-Utilities
DM 29,90* (sFr. 24,90/öS 299,-*)
Bestell-Nr. CB 024 Hilfsprogramme
DM 29,90* (sFr. 24,90/öS 299,-*)

* inkl. MwSt. Unverbindliche Preisempfehlung.

Bitte verwenden Sie für Ihre Bestellung und Überweisung die eingeheftete Postgiro-Zahlkarte, oder senden Sie uns einen Verrechnungs-Scheck mit Ihrer Bestellung. Sie erleichtern uns die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

INHALT



Speicherlandschaft

Der Teil des C64, dem er seinen Namen verdankt, ist sein 64 KByte großer Arbeitsspeicher. Doch was heißt das eigentlich? Und wie ist dieser ganze Speicher aufgebaut? In Form eines kleinen Kurses möchten wir diese und ähnliche Fragen klären, und Sie mit dem Innenleben Ihres Computers vertraut machen.

Ist es Ihnen auch so ergangen? Irgendwann stand der Commodore 64 in Ihrem Zimmer und verlangte nach Programmen. Doch woher? Software kaufen, das ist teuer; selber schreiben, dazu muß man das System kennen. Was ist denn das: ROM, CPU...?

In diesem Kurs wollen wir versuchen, Ihnen das Innenleben Ihres Commodore offenzulegen. Alle Fachausdrücke, die dabei eine Rolle spielen, sollen erklärt werden. Nach der Lektüre werden Sie in der Lage sein, weitergehende Literatur (falls Sie Blut geleckt haben) zu lesen und zu verstehen. Aber auch, wenn Sie sich auf diesen Kurs beschränken, beherrschen Sie Ihren Computer viel besser als vorher.

CPU kommt von »Central Processing Unit«, was soviel heißt wie »zentrale Arbeitseinheit«. Manche nennen sie auch liebevoll »Mikroprozessor«. Aber alle meinen damit den Verkehrspolizisten in unserem Computer, der im C64 den Namen 6510 trägt.

Ein emsiges schnellebiges Treiben findet hier statt: Daten gehen in Windeseile ein, werden bearbeitet und Antworten werden abgesandt bis an die entferntesten Adressen des C64. Dazu gibt es Registraturen, Rechenwerke, Sammelstellen, Verwaltungen, Nachrichtennetze und ein Büro, wo jemand sitzt und all das auch noch mitzählt. Und das im rasenden Tempo. Da geht's wirklich heiß her und es wird Sie nicht verwundern, zu hören, daß manche große Computer mit noch rascherem Arbeitstempo nur noch im kühlenden Bad mit flüssigem Helium arbeiten können.

Unserer CPU sieht man das alles gar nicht an. Wie ein glatter, rechteckiger schwarzer Käfer mit vielen Beinchen sieht sie aus. Erst wenn man ihr die Schale abnimmt und sie mit einem starken Mikroskop betrachtet, erkennt man schemenhaft die Struktur all dieser Einrichtungen. Die CPU versteht uns übrigens gar nicht, wenn wir mit ihr Basic sprechen. Alles muß ihr erst noch in eine spezielle Sprache übersetzt werden, die nur aus Nullen und Einsen besteht, in das sogenannte Binärsystem. Wenn man allerdings in dieser oder einer ihr sehr nahen Sprache mit der CPU zu sprechen versteht, dann erlebt man plötzlich einen Abglanz dieser wahnwitzigen Arbeitsgeschwindigkeit. Man nennt das dann in »Assembler« programmieren, genauer gesagt, in 6502-Assembler (denn 6502 ist ein naher Verwandter unserer 6510).

ROM und RAM

Bei dem Begriff ROM handelt es sich – wie so häufig – um die Abkürzung eines englischen Ausdrucks: »Read Only Memory«. Das heißt zu deutsch »Nur Lese-Speicher« und ist die Bibliothek im Computer. In diesen Speicherbereich kann nichts hineingeschrieben werden. So liegt zum Beispiel an der Adresse 65383 in unserem Commodore 64 das Betriebssystem-ROM. Wenn Sie mal mit dem Befehl »PRINT PEEK (65383)« nachsehen, was an dieser Adresse steht, dann finden Sie eine 1. Nun versuchen Sie mal, etwas dort hineinzuschreiben, beispielsweise mit »POKE 65383,255«.



Im Normalfall würde man bei einem erneuten »PRINT PEEK(65383)« die Zahl 255 erwarten. Hier aber findet man konstant die 1. ROM-Inhalte haben nämlich die angenehme Eigenschaft, immer vorhanden zu sein, auch nach dem Abschalten des Stroms. Deswegen sind alle »lebenswichtigen« Funktionen unseres Computers darin fest einprogrammiert.

Im ROM befindet sich also das Betriebssystem. Es besitzt beispielsweise eine Routine, die 60mal in der Sekunde abfragt, ob wir eine Taste gedrückt haben. Ebenfalls im ROM (aber an einer anderen Stelle) ist der Übersetzer, der unserer CPU sagt, was ein Basic-Befehl in der speziellen Binärsprache bedeutet. Diesen nennt man Interpreter. Daß Sie ein »A« auf dem Bildschirm sehen, wenn Sie eines eintippen, verdanken Sie dem Zeichen-ROM, in dem alle Buchstaben, Zahlen, Grafikzeichen und so weiter als von der CPU abrufbare Muster hinterlegt sind.

Eine andere Sorte von Speicher im Computer ist das RAM. Auch dies ist wieder eine Abkürzung eines englischen Ausdrucks (»Random Access Memory«), das heißt »Speicher mit beliebigem Zugriff«. Man kann aus diesem Speicher lesen und man kann hineinschreiben. Das RAM enthält nur flüchtige Informationen. Wenn der Computer ausgeschaltet wird, ist es wie das Auswischen der Schrift auf einer Tafel. Der Speicherinhalt ist weg. Wo finden wir das RAM im C 64? Grob gesagt, überall dort im Speicher, wo kein ROM ist. Das RAM ist der Bereich, wo wir uns als Benutzer austoben können. Alle Basic-Programme, Variablen, Felder und Strings schreiben wir ins RAM. Von Diskette oder Kassette laden wir ins RAM, ja selbst der Inhalt dessen, was auf dem Bildschirm zu sehen ist, ist RAM-Inhalt. Bestimmt haben Sie das schon bemerkt, wenn Sie durch POKE-Kommandos auf den Bildschirm geschrieben haben. Die Befehle »POKE 1024,1: POKE 55296,0« erzeugen zum Beispiel ein schwarzes »A«, weil in die erste RAM-Speicherstelle des Bildschirms der Code für A und in die erste Bildschirm-Farbspeicherstelle der Code für die Farbe Schwarz geschrieben wird. Aber nicht nur wir als Benutzer benötigen das RAM. Auch unser Computer hat viel Bedarf daran. Jede Tätigkeit erfordert von ihm das Ablegen von Notizen, zum Beispiel für irgendwelche Zwischenergebnisse oder Adressen. Deswegen ist ein großer Teil des RAM unserem normalen Zugriff entzogen. Lediglich durch POKE-Kommandos können wir der CPU dort ihre »Alleinherrschaft« streitig machen. Dafür haben wir allerdings dann auch die Folgen zu tragen.

Wir wollen uns später noch genau mit diesen ganzen Details befassen. Wie das ROM und das reservierte RAM aussieht, soll Ihnen ein kurzes Programm zeigen, das als Listing 1 abgedruckt ist:

```
1 REM **********************************   <250>
2 REM *                                *   <229>
3 REM *            D E M O  1          *   <060>
4 REM *                                *   <231>
5 REM *  HEIMO PONNATH HAMBURG 1985    *   <079>
6 REM *                                *   <233>
7 REM **********************************   <000>
8 REM                                      <151>
9 REM +++++++  HAUPTPROGRAMM  +++++++      <217>
10 PRINT CHR$(147):POKE 211,7:POKE 214,10:
   SYS 58640                               <243>
15 PRINT "SO SIEHT EIN ROM-BEREICH AUS:":F
   OR I=55296 TO 56295:POKE I,0:NEXT I     <002>
20 J=1:W=128:GOSUB 200:FOR I=40960 TO 4096
   0+1023:POKE I,PEEK(I):NEXT I            <239>
25 GET A$:IF A$="" THEN 25                 <077>
30 J=3:W=48:GOSUB 200:PRINT CHR$(147):POKE
    211,0:POKE 214,10:SYS 58640            <113>
35 PRINT "DAS IST DER VOM COMPUTER RESERVI
   ERTE RAM"                               <052>
37 FOR I=55296 TO 56295:POKE I,0:NEXT I    <214>
40 J=3:W=0:GOSUB 200                       <035>
45 GET A$:IF A$="" THEN 45                 <099>
50 POKE 56576,151:POKE 56578,63:POKE  5327
   2,21:POKE 648,4:END                     <254>
199 REM ++ UP.BILDSCHIRM VERSCHIEBEN ++    <228>
200 POKE 56576,(PEEK(56576)AND 252)OR J    <075>
205 POKE 56578,PEEK(56578)OR 3             <164>
210 POKE 53272,(PEEK(53272)AND 15)OR W     <027>
215 P=(W/16*1024+16384*(3-J))/256          <039>
220 POKE 648,P                             <145>
225 RETURN                                 <111>
```

Listing 1 zeigt, wie es im RAM/ROM aussieht

Tippen Sie es ein (dabei dürfen Sie die REM-Zeilen weglassen) und starten Sie mit »RUN«. Zunächst wird der Bildschirm leer, um dann anzukündigen, daß jetzt der Inhalt des ROM gezeigt wird. Sobald die Schrift schwarz wird, schaltet unser Computer den ROM-Inhalt auf den Bildschirm. Jedes Zeichen, das Sie dort sehen, gehört zum Programm des Basic-Interpreters und ist für uns zunächst völlig unverständlich. Das liegt daran, daß wir auf dem Bildschirm die Binärsprache der CPU als Bildschirmcode (POKE-Code) dargestellt sehen. Und das sind eben irgendwelche Zeichen mit Codes zwischen 0 und 255. Sehen Sie genau hin, dann stellen Sie fest, daß – nachdem das Bild fertig ist – keinerlei Veränderung mehr zu erkennen ist. Das ist eben das Merkmal von ROM-Inhalten. Sie bleiben, wo und wie sie sind.

Das Bild bleibt so stehen, bis Sie eine Taste drücken (bitte nicht die RUN/STOP-Taste). Danach meldet sich wieder der normale Bildschirm mit der Ankündigung, daß als nächstes der vom Computer reservierte RAM gezeigt wird. Das geschieht nach einer kurzen Zeitspanne. Da ist jetzt ordentlich was los auf dem Bildschirm! Das Flackern an einigen Stellen zeigt, daß die CPU dort gerade Eintragungen verändert.

Hoffentlich entschädigt Sie dieser Einblick in sonst verborgene Bereiche unseres C 64 dafür, daß Sie vom Programm noch kaum etwas verstehen. Seien Sie jedoch unbesorgt, im Verlauf dieses Kurses wird sich das noch ändern. Vorerst aber achten Sie darauf, daß wirklich alle Programmzeilen fehlerfrei eingegeben werden. Eine falsche Zahl könnte den Computer zum Absturz bewegen, aus dem nur noch das Aus- und wieder Einschalten rettet. Deshalb (das sollte man ohnehin immer machen): Vor dem RUN speichern.

Wir werden uns nun nach und nach soweit vorarbeiten, daß wir verstehen, was im Speicher enthalten ist.

Bits und Bytes, was sind das?

Ein Computer ist ein elektrisches Gerät. Wie alle solche Geräte kennt er zwei Zustände: Strom an und Strom aus. Jedes seiner Teile arbeitet mehr oder weniger nach diesem Prinzip. Elektrische (oder magnetische Ladung) ist vorhanden oder ist nicht vorhanden. Dazwischen gibt es nichts. Man kann einen Schalter nicht halb anschalten. Den Zustand »kein Strom« (oder keine Ladung) bezeichnet man mit Null, den anderen »Strom an« (oder Ladung vorhanden) mit Eins. Dieses nennt man die kleinste Informationseinheit oder auch »Bit«. Das Wort kommt von »binary digit«, was soviel heißt wie »binäre Ziffer«.

Für ein Bit gibt es also zwei Zustände (0 oder 1). Nehmen wir nun mal zwei Bit, dann sehen wir folgende Kombinations-Möglichkeiten:

00 1. Kombination
01 2. Kombination
10 3. Kombination
11 4. Kombination

Das sind dann $2^2 = 4$ Zustandskombinationen.

Damit lassen
3 Bit $2^3 = 8$ Varianten
4 Bit $2^4 = 16$
5 Bit $2^5 = 32$
6 Bit $2^6 = 64$
7 Bit $2^7 = 128$ und
8 Bit $2^8 = 256$
Kombinationen zu.

Nach bestimmten Rechenregeln kann man jeder Bit-Zusammenstellung auch eine Zahl zuordnen. Die zu so einer Zahl gehörende Bit-Kombination nennt man Binärzahl.

Um das Ganze übersichtlicher zu gestalten, geben wir einigen Gruppen von Bits (egal, welche Zustände bei ihnen herrschen, also ob sie nun 0 oder 1 enthalten) auch Namen. Eine Gruppe von vier Bit wird meistens ein Nibble genannt. Zwei solcher Nibbles oder acht Bit werden zusammengefaßt unter dem Begriff Byte. Da haben wir es also, das Byte.

Weiter oben haben wir festgestellt, daß man acht Bit zu 256 verschiedenen Kombinationen zusammenstellen kann. Jede Kombination entspricht einer Zahl. Deswegen kann ein Byte 256 verschiedene Zahlen enthalten, nämlich 0 bis 255. Alles, was wir durch POKE-Kommandos eingeben, geht in ein Byte (deswegen meldet der Computer einen »ILLEGAL QUANTITY ERROR«, wenn wir andere Zahlen als solche zwischen 0 und 255 einzuPOKEn versuchen) und alles, was wir durch PEEK erhalten können, ist ebenfalls der Inhalt eines Byte. Glücklicherweise übernimmt auch hier der Interpreter die Übersetzung in das System der Binärzahlen und umgekehrt.

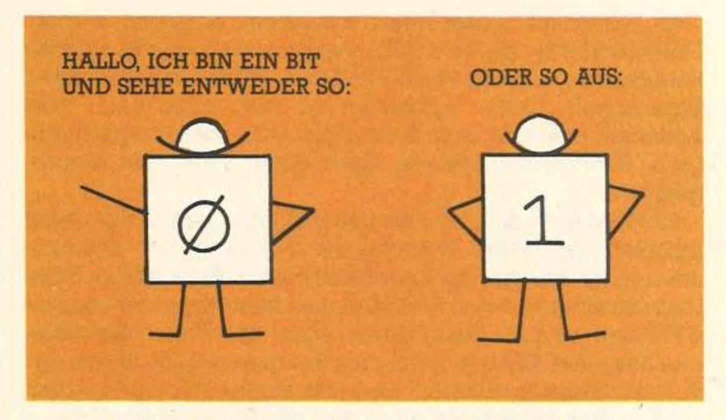
In unseren Computer passen genau (ohne die ROM-Bausteine) 65536 solche Bytes hinein. Das sind schon wieder unüberschaubar viele, weswegen man auch hier einige Einteilungen vornimmt.

Pages und »Kilo«-Bytes, so viele Fremdwörter

»Kilo«-Byte sind eigentlich keine solchen. Kilo, abgekürzt »k«, steht immer für 1 000 mal etwas. Also 1 km sind 1000 mal 1 Meter, 1 kg sind 1000 Gramm und so weiter. Bei den so beliebten »Kilobyte« ist das anders: So nennen nämlich viele in der Computer-Umgangssprache eine Menge von 2^{10} Bytes. Und 2^{10} sind 1024. Deswegen kürzt man diesen Begriff auch nicht mit dem kleinen »k«, sondern mit dem großen »K« ab. Und deswegen ist der Ausdruck »Kilobyte« falsch. Exakt spricht man es »Ka-Byte« aus. Nochmal also: 1 000 Byte = 1 kByte (»Kilobyte«), aber 1 024 Byte = 1 KByte (»Ka-Byte«).

Und wenn Sie nun mal 65 536 durch 1 024 teilen, dann sehen Sie, daß in unseren Computer 64 KByte hineinpassen. Deswegen heißt er auch Commodore 64. Wenn jemand sagt, ein Programm sei 28 K lang, dann sollte er wissen, daß er eigentlich meint, es sei 28 KByte lang, was 28 x 1024 = 28 672 Byte entspricht oder aber 28,672 kByte.

1 024 Byte sind immer noch etwas unhandlich. Wenn wir diese Menge durch 4 teilen, dann erhalten wir 256 Byte. Genau diese Anzahl bildet eine »Page«, was von dem englischen Wort für »Seite« (wie eine Buchseite) herrührt. Page ist allerdings weniger die Bezeichnung für genau 256 Byte (sagt man aber auch manchmal), sondern ein Ordnungsprinzip wird damit ausgedrückt. Wenn man nämlich alle Bytes im Computer durchnumeriert (von 0 bis 65 535) und dabei



immer so zählt, daß jeweils 256 davon eine Page bilden, dann erhält man 256 Pages zu je 256 Byte. 256 x 256 ergibt dann wieder 65 536. Diese Pages tragen dann – wie in einem Buch – Seitenzahlen, und mit Page 0 (Adressen 0 bis 255), der sogenannten »Zero«-Page, fängt unser Computer an.

Diese Zeropage ist einer von den RAM-Bereichen, die der Computer für seine Notizen reserviert hat. Das obere Viertel unseres Bildschirmes hat diese Page vorhin beim Listing 1 wiedergegeben und ein Zeichen dabei entsprach genau einem Byte.

Beim Wort Vektor denken vielleicht einige von Ihnen mit mehr oder weniger leichtem Schaudern an den Mathematik- oder Physik-Unterricht in der Schule. Damit hat der Vektor, den wir im Computer finden, aber wenig gemein.

Im Abschnitt über die CPU haben wir gesehen, daß dort Daten eingehen, verarbeitet und schließlich bis in die entferntesten RAM-Bereiche unseres Computers gesandt werden. Außerdem wissen wir (oder Sie haben das jedenfalls inzwischen sicher bemerkt), daß das Byte genau der »Happen« ist, der dem Computer – also auch der CPU – mundgerecht verabreicht werden kann (beispielsweise in POKE-Befehlen). Drittens – und jetzt kommt das Problem – gibt es 65 536 Byte im C 64. Die CPU muß oft Daten an eine Speicherzelle senden, deren Adresse höher ist als 255 (also das Maximum, das mit einem Byte darstellbar ist). Weil aber ein 256-Byte-Computer (0,25 KByte) doch ein wenig kläglich wäre, hat man sich da etwas einfallen lassen. Die Adressen werden in zwei Bytes aufgeteilt und einzeln gelesen. Durch eine bestimmte Technik des Assembler-Programms werden sie zu einer Zahl größer als 256 verkoppelt. Erinnern Sie sich noch daran, daß mit acht Bit (also 1 Byte) 2^8 = 256 Kombinationen erreichbar waren? 2 Byte – also 16 Bit – sind dann demzufolge 2^{16} Kombinationen, was exakt 65 536 Möglichkeiten eröffnet.

Wie kann man nun herausfinden, wie beispielsweise die Zahl 50 000 in 2 Byte zerlegt wird, so daß der Computer sie richtig verwendet? Natürlich haben auch diese 2 Byte wieder Namen. Das eine heißt MSB und das andere LSB. SB kommt dabei immer von »Significant Byte«, was soviel bedeutet wie »wichtiges Byte«. »M« steht für »Most«, also »am meisten«

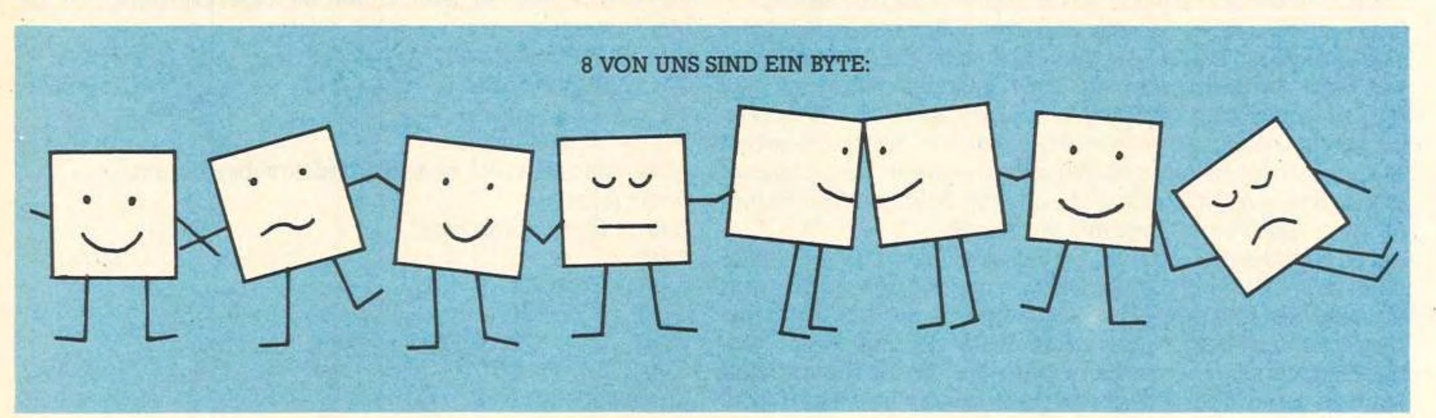




Bild 1. Die Speicherstadt

und »L« ist die Abkürzung für »Least«, also »am wenigsten«. Wir haben also ein höherwertiges und ein niederwertiges Byte.

Das MSB gibt die Pagenummer der Adresse an. Man erhält diese Nummer durch Teilen der Adresse mit 256 und Weglassen des Restes. Nehmen wir zum Beispiel die Zahl 50 000, dann rechnet man:

MSB = 50 000/256 = 195 Rest 80

Dieser Rest ist das LSB. 50 000 ist dann das 80. Byte auf Page 195. Allgemein kann man sowas ja seinen Computer rechnen lassen. Wenn also die Zahl Z in ein MSB und ein LSB aufgespalten werden soll, erreicht man das mit dieser Programmzeile: »MSB=INT(Z/256):LSB=Z-256*MSB«

Anders herum: Wenn eine Zahl als MSB und LSB vorliegt, kann man sie so berechnen:

Z=256 * MSB + LSB

Wozu das alles? Solche Byte-Paare nennt man Vektoren und die werden vom Computer im RAM gespeichert. Und wenn etwas im RAM steht, kann man es durch POKE-Befehle ändern, vorausgesetzt man weiß, wie so ein Vektor umgerechnet werden muß.

In solchen Vektoren sind allerhand wichtige und interessante Dinge zu finden. Im weiteren Verlauf dieses Kurses werden wir uns damit noch beschäftigen. Man kann zum Beispiel durch Umschreiben eines Vektors Speicherbereiche schützen und vieles mehr.

Die Helfer der CPU

Auch die tüchtigste Zentrale – wie unser Mikroprozessor 6510 – ist ohne die wirkungsvolle Mitarbeit weiterer Organe hilflos. Das können Sie leicht feststellen, wenn Sie diese alle abschalten. Wir haben dann nur noch die CPU und 64 KByte RAM vor uns. Bevor Sie aber den dazu nötigen Befehl »POKE1,PEEK(1)AND248« eingeben, sollten Sie alle Daten und Programme auf Kassette oder Diskette speichern, denn danach ist unser Computer scheintot, und nur noch durch Aus- und wieder Einschalten erwecken wir ihn zum gewohnten Leben.

Was macht uns den Verkehr mit unserem Computer überhaupt erst möglich? Weiter oben haben wir einiges davon schon erwähnt. Als 8 KByte umfassenden Baustein haben wir da zunächst den Basic-Interpreter, der alle unsere Wünsche, die wir in Basic dem Computer mitteilen, in die der CPU allein verständliche Sprache aus Nullen und Einsen übersetzt. Es handelt sich dabei um ein ROM, in dem ein Programm in Maschinensprache unveränderbar gespeichert ist. Man nennt sowas auch Firmware, weil es Software ist, die vom Hersteller fest installiert wurde.

Zur Begriffserklärung: Hardware ist alles, was man – vereinfacht ausgedrückt – am Computer anfassen kann, Chips, Platinen und so weiter. Software nennt man die Programme.

Harte Sache

Ein zweiter, ebenfalls 8 KByte umfassender ROM-Baustein enthält das Betriebssystem unseres Commodore, Kernel genannt. Auch hier handelt es sich um Firmware und eine ihrer Aufgaben, nämlich die Abfrage der Tastatur, haben wir weiter oben schon erwähnt. Andere sind beispielsweise die Organisation der Datenübergabe an die Datasette oder den Bildschirm, das Weiterstellen der internen Uhren, das Initialisieren nach dem Einschalten und vieles mehr.

An weiterer Firmware finden wir in unserem C64 das Zeichen-ROM, welches, auf 4 KByte verteilt, alle Zeichen, die unser Computer darstellen kann, als Muster enthält. Aus diesem Speicher werden dann beispielsweise Buchstaben durch ein im Betriebssystem enthaltenes Programm herauskopiert und an eine festgelegte Bildschirmadresse übertragen.

Zur Ausführung einer solchen Übertragung bedarf es nicht nur der Programme beziehungsweise der Daten, die wir in den drei beschriebenen Bausteinen als Firmware kennengelernt haben, sondern auch bestimmter Instrumente, die alle damit zusammenhängenden Anweisungen in die Tat umsetzen. Das geschieht im Commodore 64 durch vier Bausteine: CIA 1, CIA 2, SID und VIC-II-Chip.

Fangen wir mit dem letzten an: VIC (so heißt übrigens im englischen Sprachraum der VC 20, was auf die Bedeutung dieses Bausteins hinweist) kommt von »Video Interface Controller«. Dieser Baustein regelt den Verkehr unseres Computers mit dem Bildschirm. Er beansprucht genau 1 KByte an Speicherplatz und verfügt über 47 Register, deren Inhalte seine Tätigkeit steuern. Für alle, die grafische Datenverarbeitung betreiben möchten, ist dieser Chip der Schlüssel dazu.

SID ist die Abkürzung von »Sound Interface Device«. Der akustische Verkehr mit der Außenwelt wird durch diesen ebenfalls 1 KByte großen Baustein gemanaget. 29 Register erlauben hier die Funktionssteuerung, die den Commodore 64 zum Synthesizer oder – mit einigen Tricks – zum Sprachausgabegerät ummodelt.

CIA steht für »Complex Interface Adapter«. CIA 1 und CIA 2 sind zwei identische Bausteine, die aber unterschiedliche Aufgaben zu erfüllen haben. Beide beanspruchen je 256 Byte Speicherplatz und sind über ihre 16 Register zu beeinflussen. Sie führen vor allem die Ein- und Ausgabeoperationen aus. Dabei ist der CIA 1 zuständig für Tastatur, Joystick, Lichtgriffel und Paddles, während sich der CIA 2 um den User-Port, die RS232C-Schnittstelle, den seriellen Ausgang und die interne Speicherstruktur zu kümmern hat. Beide haben außerdem Uhrenfunktionen und regeln das sogenannte Interrupthandling (dazu kommen wir später).

Stellen wir also fest, daß alle diese Helfer unserer CPU insgesamt 22,5 KByte Speicherplatz belegen. Dazu kommen noch 1 KByte Bildschirmfarbspeicher und 0,5 KByte, die von Commodore für Erweiterungen freigehalten werden. Wenn Sie den C64 einschalten, dann meldet er sich unter anderem mit »64 K RAM SYSTEM«. Da kann aber doch etwas nicht stimmen, werden Sie sagen, wenn Sie bis hierher aufmerksam mitgelesen haben. Etwas weiter oben haben wir festgestellt, daß unsere Zentraleinheit Zugriff zu allen – auch den entferntesten – Adressen hat. Dazu verwendet sie den Trick mit dem Aufteilen einer »Anschrift« auf zwei Byte und konnte so exakt 65536 Adressaten erreichen, also 64 KByte. Nach Adam Riese summieren sich aber 64 KByte

RAM und (etwas vereinfacht) 22,5 KByte ROM zu insgesamt 86,5 KByte Speicherraum, den es zu adressieren gilt. Das ist mittels 2 Byte Anschriftentext aber nicht möglich. Diesen scheinbaren Widerspruch werden wir gleich auflösen. Dazu müssen wir aber unseren Speicher noch etwas genauer untersuchen.

Die Speicherstadt

Stellen Sie sich eine lange Straße vor mit 65536 aneinandergereihten Häusern (von Hausnummer 0 bis Hausnummer 65535, wie in Bild 1).

Dies entspricht unserem Speicher. Jedes Haus (Byte) ist ebenerdig und hat acht Zimmer (Bits). Wie eine Stadt in Stadtteile unterteilt ist, finden wir in dieser Speicherstadt die Einteilung in Pages. Ähnlich wie es in Städten ein Handwerkerviertel und ein Geschäftsviertel und so weiter gibt, sind auch hier manchen Pages spezielle Aufgaben zugeteilt. Die wichtigste davon ist die Zeropage, auf der sich die CPU beziehungsweise das Betriebssystem Notizen machen. Auch die Pages 1 bis 3 (also bis Adresse 1023) dienen ähnlichen Zwecken. Ab Page 4 bis inklusive Page 7 liegt der Bildschirmspeicher unseres Computers. Er entspricht genau dem, was auf dem Fernsehbild zu sehen ist. Jedes Zeichen wird dabei durch einen POKE-Code vertreten. Die Zuordnung der einzelnen Adressen zu den Bildschirmpositionen kann man aus dem Handbuch (Seite 138) entnehmen, ebenso wie die POKE-Codes (Seite 133). Packen wir also in Adresse 1024 eine 1 hinein durch »POKE 1024,1«, dann erscheint in der linken oberen Bildschirmecke ein »A«. Bei Ihnen erscheint kein »A«? Dann fahren Sie mal mit dem Cursor an die Stelle und Sie erkennen den Buchstaben. Den Commodore 64 gibt es momentan mit mindestens zwei verschiedenen Versionen des Betriebssystems. Bei der älteren muß man außer dem Bildschirmcode in den Bildschirmspeicher auch noch einen Farbcode in die entsprechende Bildschirmfarbspeicherstelle geben. Diese kann man ebenfalls dem Handbuch entnehmen (Seite 139). Hier braucht man also noch den Befehl »POKE 55296,1«, um ein weißes »A« zu erzeugen. Die neuere Version macht den Farbcode-POKE überflüssig. Nur wenn wir eine andere Farbe als die vorgegebene möchten, müssen wir den neuen Farbcode in den Bildschirmspeicher POKEn.

Der Bildschirmspeicher erfordert genau 25 * 40 = 1 000 Byte. Von den 1 024 Byte (4 Pages) sind also noch 24 Byte frei, die teilweise Verwendung finden als Sprite-Zeiger. Doch dazu kommen wir erst später. Ab Page 8 (Adresse 2048) haben wir volle Verfügungsgewalt über den Speicher für Basic-Programme und Daten.

Etwas Neues passiert ab Adresse 40960, dem Ende unseres Basic-Speichers. Von dieser »Hausnummer« an, bis 49151, haben die Gebäude der Speicherstadt eine zusätzliche erste Etage. Zu ebener Erde liegt weiterhin RAM vor, im ersten Stock aber ROM, und zwar der 8 KByte große Basic-Interpreter (siehe Bild 2).

In den nächsten 4 KByte finden wir wieder nur RAM. Dieser Bereich von 49152 bis 53247 wird häufig für Maschinenprogramme genutzt, weil hier nicht die Gefahr des unabsichtlichen Überschreibens durch Basic-Programme besteht. Ab 53248 sind die »Byte-Häuser« sogar mit zwei Etagen versehen. Im Erdgeschoß liegt weiterhin RAM, in der ersten Etage sind die Ein- und Ausgabe-Bausteine angesiedelt und oben im zweiten Stockwerk breitet sich das Zeichen-ROM aus. Die Belegung im ersten Geschoß durch die Ein- und Ausgabe-Bausteine ist in Bild 3 zu sehen.

Dabei belegt der VIC-II-Chip die Adressen von 53248 bis 54271, der SID-Chip die von 54272 bis 55295, der Bildschirmfarbspeicher die von 55296 bis 56319, der CIA 1 liegt von 56320 bis 56575, der CIA 2 von 56576 bis 56831 und

Bild 2. Die Speicherarchitektur des C64

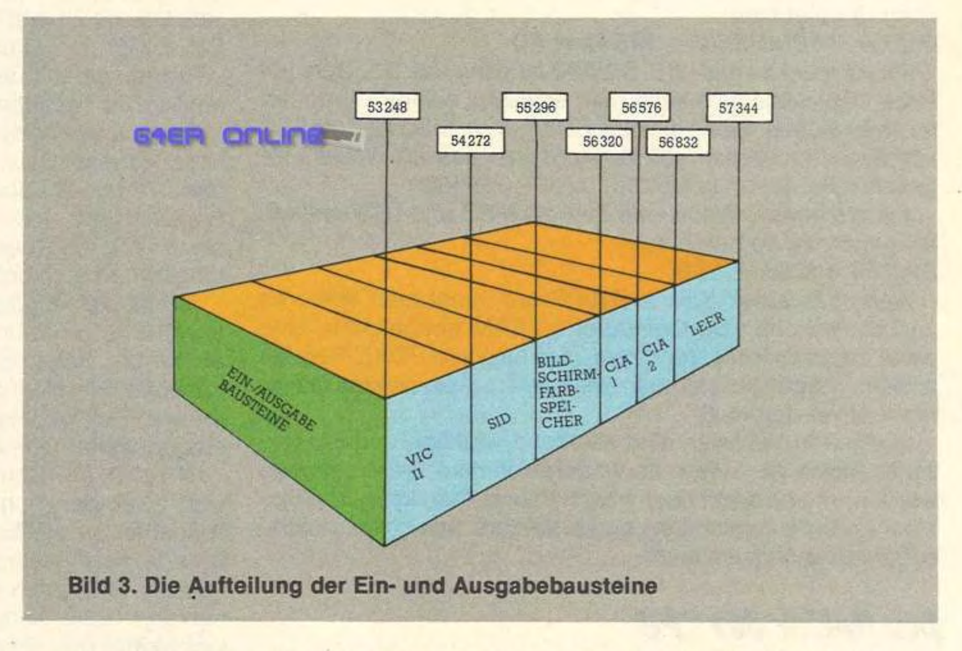


Bild 3. Die Aufteilung der Ein- und Ausgabebausteine

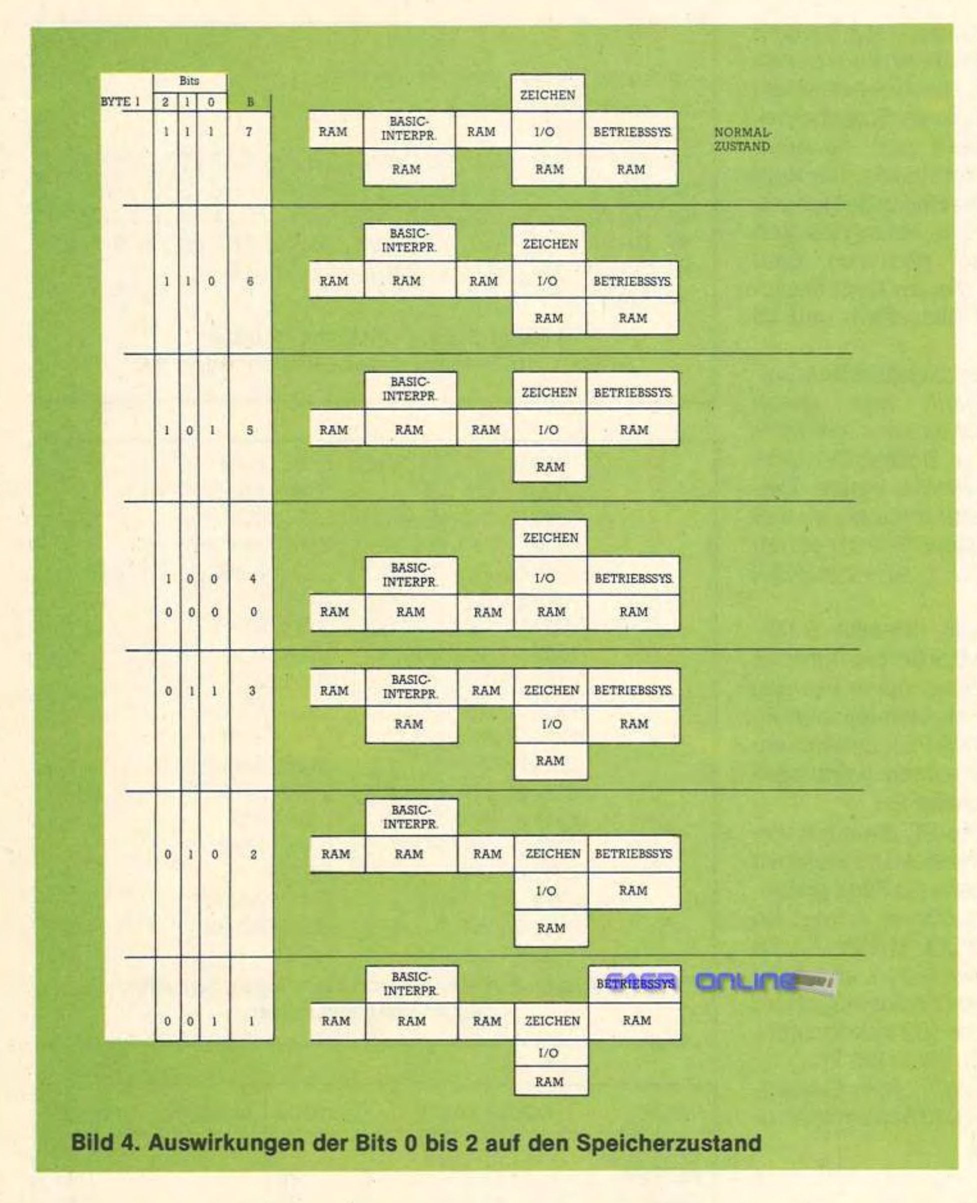


Bild 4. Auswirkungen der Bits 0 bis 2 auf den Speicherzustand

der Bereich von 56832 bis 57343 ist leer. Ab Adresse 57344 bis zum Speicherende haben wir es dann wieder mit einstöckigen »Byte-Häusern« zu tun, in deren Erdgeschoß RAM und in der ersten Etage das Betriebssystem-ROM wohnt.

Jetzt können wir an die Lösung des Rätsels gehen, wieso unsere CPU anscheinend mehr als 64 KByte an Hausnummern adressieren kann. Wie Sie bei der Wanderung durch die Speicherstadt bemerkt haben werden, sind es ja nur 65536 Häuser, die zu adressieren sind. Die Frage ist jetzt nur noch, woher die CPU weiß, in welche Etage bei den mehrstöckigen Byte-Gebäuden eine Nachricht gelangen soll, beziehungsweise aus welcher Etage eine Meldung zu holen ist. Dazu verwendet unser Computer die Speicherstelle 1. In drei von den acht Bit-Zimmern dieses Byte-Hauses liegt die Information, welche Etage in welchen Speicherbereich gerade zugänglich ist. Es gibt für drei Bit (Bit 0 bis 2) ja $2^3 = 8$ Kombinationsmöglichkeiten. In Bild 4 sehen Sie die Auswirkung der verschiedenen Inhalte von Bit 0, 1 und 2 dieser Speicherstelle.

In Bild 4 ist auch jeweils der Dezimalwert der Bits 0 bis 2 angegeben (in Spalte B). Durch Eingabe des Basic-Befehles »`POKE 1, (PEEK (1) AND 248) OR B`«, wobei B dann den entsprechenden Wert annimmt, kann der Speicherzustand gesteuert werden. Allerdings gilt es dabei noch ein paar Feinheiten zu beachten, damit der Computer nicht abstürzt. Auch kann man diese veränderten Speicherkonstellationen nur selten von Basic aus sinnvoll nutzen. Die unterschiedliche Belegung der Speicherstelle 1 wird auch vom Betriebssystem wahrgenommen und dadurch ist unsere CPU in der Lage, festzustellen, welche Etage gerade für den Zugriff offensteht. Mit dieser Lösung des Rätsels sind wir schon mitten in der Zeropage (Adressen 0 bis 255) gelandet, die uns nun weiter beschäftigen soll.

Wenn Sie das Handbuch zum Commodore 64 auf Seite 160 aufschlagen, finden Sie eine Liste der Belegung der Zeropage und ab Seite 163 eine der Pages 1 bis 3. Deshalb soll hier nicht noch mal diese Tabelle abgedruckt werden. Außer einzelnen Werten und Vektoren finden sich hier sogar ganze Maschinenprogramme.

Wir werden hier nur diejenigen Adressen genauer untersuchen, die Verwendungsmöglichkeiten für uns im Rahmen von Basic bieten. Außer der schon behandelten Speicherstelle 1 sind die Adressen 43 bis 56 von sehr großem Interesse. Hier wird nämlich der Basic-Speicher organisiert. Zunächst soll uns nur der Vektor 43/44 beschäftigen. Um die anderen aus diesem Bereich zu verstehen, brauchen wir noch mehr Informationen über den Aufbau von Basic-Programmen im Speicher. 43/44 ist der Vektor, der auf den Anfang eines Basic-Programmes zeigt. Wenn Sie mit »`PRINT PEEK (43), PEEK (44)`« den Computer danach fragen, dann antwortet er im Normalfall mit den Angaben »1« und »8«. Das sind – in dieser Reihenfolge – das LSB und das MSB der Startadresse, und wenn Sie sich die Umrechnungsformel: $Z = 256 * MSB + LSB$ zunutze machen, dann erhalten Sie: $Z = 256 * 8 + 1 = 2049$. Das ist eine Speicherstelle weiter, als die, die wir nach dem Durchgehen des gesamten Speichers erwartet hatten. In der Speicherstelle 2048 liegt eine Null, womit dem Interpreter der Anfang noch etwas deutlicher gemacht wird.

Wie wir weiter oben schon erwähnt haben, kann man solche Vektoren im RAM nicht nur lesen, sondern auch verändern. Das soll an einem Beispiel demonstriert werden: Nehmen wir an, unser Basic-Speicher-Anfang soll verlegt werden nach Speicherstelle 12288. Dann müssen wir zunächst die Null von 2048 nach 12288 verschieben, und zwar mit »`POKE 12288,0`«. Als nächstes berechnen wir das LSB und das MSB von 12289: MSB=INT (12289/256)=INT (48.003906) =48 und LSB=12289-256*48=1. Nun POKEn wir LSB und MSB des neuen Basic-Anfanges nach 43 und 44 (»`POKE 43,1:Poke 44,48`«). Damit zum einen der so definierte Speicherraum leergefegt wird und zum anderen auch einige andere Vektoren in die richtige Stellung kommen, wird abschließend noch NEW eingegeben (»`NEW`«).

Wozu benötigt man so etwas? Sehr häufig verwendet man Basic- und Maschinencode-Programme gemeinsam. Sei es, daß man nur eine kleine Routine – beispielsweise zum Sortieren – in Maschinensprache vom Basic-Programm her aufruft oder daß man ein Basic-Rahmenprogramm verwendet, um ein komplexes Maschinencode-Programm zu bedienen, immer braucht man einen geschützten Platz für das Maschinen-

programm. Zwar kennen Sie ja inzwischen den RAM-Bereich von 49152 an, der für viele solche Zwecke dient, aber genau darin liegt oft das Problem. Wenn Sie nun beispielsweise ein großes Assemblerprogramm in diesem oberen Speicherbereich ständig präsent haben wollen, dann muß die neue Assembler-Routine ein anderes Plätzchen finden. Ich lege sie in solchen Fällen meist an das Ende des Basic-Speichers. Man kann sie aber ebensogut ganz an den Anfang packen und den Basic-Programm-Start dahinter plazieren. Eine andere Verwendung sind Sprite-Daten, die vor dem Basic-Programm liegen sollen oder ein Grafik-Bildschirm und so weiter.

In den Speicherzellen 59/60 wird immer die letzte bearbeitete Zeilennummer festgehalten. Wenn man durch RUN/STOP ein Programm anhält, meldet unser Computer diese Zahl in der Mitteilung »BREAK IN...«. Sollten Sie nach einer solchen Unterbrechung beispielsweise einige Zwischenrechnungen im Direktmodus ausgeführt haben, so daß die Zeilennummer nicht mehr auf dem Bildschirm zu sehen ist, dann können Sie sie durch »`PRINT PEEK (59) + 256*PEEK (60)`« wieder nachlesen.

In den Bytes 63/64 wird die jeweils aktuelle DATA-Zeilennummer gespeichert, wohingegen 65/66 die Adresse des aktuellen DATA-Elementes enthält. Eine kleine Routine für »RESTORE Zeilennummer« finden Sie als kleines aber feines Programm (Listing 2 und 3). Mit »SYS 49152, Zeilennummer« können Sie nun diesen Befehl einsetzen (vorausgesetzt, das Programm ist im Speicher enthalten).

In Byte 144 finden wir die Statusvariable ST, die auch von Basic her abgefragt werden kann. Speicherstelle 152 enthält die Anzahl offener Files. Man darf höchstens 10 Files gleichzeitig offenhalten. Versucht man mehr zu öffnen, erfolgt ein Programmabbruch und die Meldung »TOO MANY FILES ERROR« wird ausgegeben. Um das zu vermeiden, empfiehlt es sich in manchen Programmen, die dieses Risiko eingehen, zuvor eine Abfrage des Inhaltes von Byte 152 durchzuführen, zum Beispiel mit »`IF PEEK (152)=10 THEN PRINT...`«.

Byte 157 trifft die Unterscheidung, ob sich unser Computer gerade im Direktmodus (dann enthält es 128) oder im Programmodus (dann enthält es 0) befindet.

Viele nützliche Adressen

Ein Unterschied, der ins Auge fällt, ist das Verhalten des Computers bei Systemmeldungen. Im Programmodus werden diese unterdrückt (kein SEARCHING oder LOADING und so weiter). Wenn Sie also wünschen, daß auch innerhalb eines Programmes diese Meldungen auftreten, dann stellen Sie durch »POKE 157,128« diese Meldungen an oder wenn Sie die Meldungen im Direktmodus stören, durch »POKE 157,0« aus. Speicherstelle 184 enthält die Nummer des zuletzt geöffneten Files.

In Speicherstelle 185 findet man – in modifizierter Form – die aktuelle Sekundäradresse, die in diesem Fall den Kanal angibt. Die Veränderung der eigentlichen Sekundäradresse kann durch folgende Abfrage aufgefangen werden: »Sekundäradresse = PEEK (185) AND 159«. Die Geräteadresse des zuletzt eröffneten Files findet man in Speicherzelle 186. Im Byte 198 befindet sich die Anzahl der gültigen Zeichen im Tastaturpuffer. Das ist ein Speicherbereich, der maximal zehn Tastendrucke zwischenspeichern kann, wenn sie aus irgendwelchen Gründen nicht sofort verarbeitbar sind. Man kann diese Anzahl variieren, sollte aber nie größere Zahlen als 10 eingeben, weil dadurch Störungen des Systems ausgelöst werden könnten. Einige Anwendungen werden wir zusammen mit dem Tastaturpuffer behandeln. Mit dem WAIT-Befehl läßt sich das umständliche »GETA$:IFA$= ""THEN...« in folgender Weise vereinfachen: »WAIT 198,1: POKE 198,0«

```
10 FOR I=49152 TO 49195
20 READ A:POKE I,A
30 NEXT I
40 DATA 32,121,0,201,156,240,3,76,8,175
50 DATA 32,115,0,32,138,173,32,247,183
60 DATA 165,20,133,63,165,21,133,64,32
70 DATA 19,166,32,248,168,165,20,164,96
80 DATA 56,233,1,32,36,168,60
90 END
```

Listing 2. Eine nützliche Routine: »RESTORE Zeilennummer« (Basic-Programm)

```
C000 20 79 00  JSR $0079
C003 C9 2C     CMP #$2C
C005 F0 03     BEQ $C00A
C007 4C 08 AF  JMP $AF08
C00A 20 73 00  JSR $0073
C00D 20 8A AD  JSR $AD8A
C010 20 F7 B7  JSR $B7F7
C013 A5 14     LDA $14
C015 85 3F     STA $3F
C017 A5 15     LDA $15
C019 85 40     STA $40
C01B 20 13 A6  JSR $A613
C01E 20 F8 A8  JSR $A8F8
C021 A5 5F     LDA $5F
C023 A4 60     LDY $60
C025 38        SEC
C026 E9 01     SBC #$01
C028 20 24 A8  JSR $A824
C02B 60        RTS
```

Listing 3. Für Kenner von Maschinen-Sprache: Das Assembler-Listing

TASTE	CODE	TASTE	CODE	TASTE	CODE
DEL	0	T	22	–	43
RETURN	1	X	23	.	44
CURSOR ⇄	2	7	24	:	45
f7	3	Y	25	@	46
f1	4	G	26	,	47
f3	5	8	27	£	48
f5	6	B	28	*	49
CURSOR ↑↓	7	H	29	;	50
3	8	U	30	HOME	51
W	9	V	31	=	53
A	10	9	32	↑	54
4	11	I	33	/	55
Z	12	J	34	1	56
S	13	0	35	←	57
E	14	M	36	2	59
5	16	K	37	SPACE	60
R	17	O	38	Q	62
D	18	N	39	STOP	63
6	19	+	40	keine Taste	64
C	20	P	41		
F	21	L	42		

Tabelle 1. Codes, die in Speicherzelle 203 bei Tastendruck zu finden sind

Speicherstelle 199 enthält ein Flag, das anzeigt, ob revers oder normal gedruckt wird. Ist der Inhalt von 199 eine 1, dann ist der Reversmodus an-, bei 0 ausgeschaltet. Speicherstelle 203 enthält einen Index, der entsprechend der gerade gedrückten Taste auf den dazugehörigen Ort der Tastaturdecodierungstabelle weist. Dieser Index ist weder mit dem POKE-Code noch mit dem Commodore-ASCII identisch.

Durch »PEEK (203)« kann dieser Wert überprüft und dann darauf reagiert werden. Welche Taste zu welchem Code gehört, können Sie aus der Tabelle 1 entnehmen.

Die Speicherstellen 204 und 207 sind verwendbar, um auch bei GET-Abfragen einen Cursor auftreten zu lassen. In Byte 204 wird durch den Wert 0 das Blinken des Cursors ein-, durch 1 ausgeschaltet. Weil man aber nicht genau vorhersagen kann, ob beim Ausschalten gerade der Cursor sichtbar war – und dieser dann als heller Block erhalten bleiben würde – kann man ein Flag in 207 auf 0 setzen, um diesen Block verschwinden zu lassen. Das sollte vor dem Ausschalten des Cursorblinkens geschehen. Im folgenden Programmbeispiel ist so eine Sequenz gezeigt:

```
10 PRINT CHR$ (147)"BITTE EINGABE!";
20 POKE 204,0:POKE 198,0
30 WAIT 198,1:GETA$
40 POKE 207,0:POKE 204,1
50 ....
```

Die Speicherstellen 211 und 214 enthalten die aktuelle Cursor-Position. 211 gibt dabei die Spalte (0 bis 39), 214 die Zeile (0 bis 24) an. In Zusammenhang mit einer Betriebssystemroutine kann man aber auch in diese Speicherzellen Werte eingeben, um den Cursor an bestimmte Positionen zu setzen. In meinen Programmen habe ich zu diesem Zwecke immer ein kleines Unterprogramm eingebaut:

```
10 POKE 211,SP:POKE 214,Z:SYS 58640:RETURN
```

Das benütze ich dann immer mit Angabe des Spalten(SP)- und Zeilenwertes (Z) und erspare mir damit den Wust an Cursorsteuerbefehlen oder CHR$-Anweisungen.

Interessant ist auch der Vektor 243/244, der die zur aktuellen Cursor-Position gehörige Bildschirmfarbspeicherzelle angibt. Mit unserer Formel kann diese dann einfach berechnet werden: »Farbzelle = PEEK(243) + 256*PEEK (244)«.

Damit wären wir am Ende der Zeropage angelangt. Lediglich die wichtigen Adressen 43 bis 56 bedürfen noch einer genaueren Erklärung: diese haben nämlich mit der Speicherung eines Basic-Programms und aller damit verbundenen Größen zu tun.

Die Speicherstellen 43 bis 56 enthalten Notizen unseres Computers zur Verwaltung von Basic-Programmen. Wir wollen uns das einmal genau ansehen. Schalten Sie bitte Ihren 64 aus und wieder an, damit wir den Speicher im Grundzustand vor uns haben. Dann geben Sie ein:
»`PRINT PEEK (43), PEEK (44)`«. Auf dem Bildschirm steht als Antwort: »1   8«.

Die Speicherstellen 43 und 44 bilden einen Vektor, der auf den Beginn des Basic-Speichers zeigt. Die Zahlen sagen uns, daß wir es mit der ersten Speicherstelle der Page 8 zu tun haben, denn mit der Formel
»ERGEBNIS = 256*MSB+LSB = 256*8 + 1 = 2049«
ergibt sich 2049. Aus der Speicheraufteilung des C64 ist uns 2048 noch als Basic-Startadresse bekannt. In dieser Speicherstelle steht eine Null, so daß das erste Basic-Programm-Byte tatsächlich erst in 2049 zu finden ist.

Jetzt machen wir uns die Sache etwas bequemer. Löschen Sie den Bildschirm und geben Sie dann im Direktmodus – also ohne Programmzeilennummer – ein:

```
A=45:PRINT PEEK(A),PEEK(A+1),PEEK(A)+256*PEEK(A+1)
```

Nach dem Return finden wir auf dem Bildschirm:
»3   8   2051«.

Das ist das Ende des Basic-Programms und gleichzeitig der Anfang des Variablenspeichers. Sie werden sich darüber wundern, daß wir ja gar kein Programm im Speicher stehen haben und trotzdem 2 Byte verbraucht worden sind. Dieses Rätsel werden wir lösen, sobald wir uns näher mit der Struktur eines Basic-Programms im Speicher befassen.

Wie Sie sehen, fangen die Variablen direkt hinter dem Basic-Programm an. Hier findet man die sogenannten einfachen Variablen, wovon es vier Typen gibt:

Normale Gleitkommavariablen	zum Beispiel A
Integer-Variablen	zum Beispiel B%
String-Variablen	zum Beispiel C$
und etwas aus dem Rahmen fallend, Funktionen	zum Beispiel FND(X)

Übrigens liegt in der Tatsache, daß die Variablen direkt hinter dem Programm zu finden sind, auch der Grund für ein zunächst etwas unverständliches Verhalten unseres Computers: Wenn Sie schon einmal ein Programm durch »STOP« oder durch die »RUN/STOP-RESTORE«-Tasten angehalten und danach eine Zeile oder Anweisung geändert haben, konnten Sie es nicht mehr mit »CONT« weiterlaufen lassen. Weshalb? Weil durch die Programmänderung eine Verlängerung stattgefunden haben könnte, die die ersten Variablen oder Teile davon unter Umständen überschrieben hätte. Deswegen ist dieses »CONT« vorsichtshalber gesperrt worden. Damit schließt man eventuell schwer festzustellende Fehler aus.

So verwaltet der C64 Basic-Programme

Auf Ihrem Bildschirm steht hoffentlich noch die vorhin eingegebene Anweisung im Direktmodus. Dann fahren Sie jetzt mit dem Cursor hoch und löschen Sie die Ergebnisse. Dann fahren Sie in die Zeile, in der das A definiert wurde. Ersetzen Sie die Zahl 45 durch 47 und drücken Sie RETURN. Auf dem Bildschirm finden Sie nun:
»10   8   2058«.

Der Vektor 47/48 weist auf das Ende der Tabelle der einfachen Variablen und gleichzeitig auf den Anfang der indizierten Variablen, welche man häufig auch Arrays oder Felder nennt. Eigentlich sollte im Leerzustand dieser Zeiger auch auf die Speicherstelle 2051 deuten. Wir haben aber in unserer Direktanweisung eine Variable A definiert. Die ist nun in die Variablentabelle eingetragen worden und verbraucht – wie wir sehen – genau 7 Byte Speicherplatz. Zwar werden wir uns später noch mit dem Aussehen der Variablen-Einträge befassen, wir können uns aber schon merken, daß jede Variable diese 7 Byte Speicherplatz beansprucht. Das gilt auch für die Integervariablen, von denen sich die oft gehörte Annahme, sie würden weniger Speicherplatz verbrauchen, als eine Ente herausstellt. Setzen Sie doch mal in unserer Direktmoduszeile anstelle von »A« ein »A%« ein: Das Ergebnis verändert sich nicht.

Aus der Tatsache, daß die Arrays direkt im Anschluß an die einfachen Variablen stehen, kann man eine Verhaltens-Maßregel für den Programmierer ablesen. Nehmen wir einmal an, die einfachen Variablen werden in einem Basic-Programm nicht eigens definiert, sondern immer automatisch dann, wenn sie gebraucht werden. Außerdem werden Arrays verwendet. Kommt also das Programm im Verlauf der Abarbeitung beispielweise an die Variable N, dann wird dafür ein Eintrag in die Variablenliste (Anfang ab Vektor 45/46 bis Ende bei Vektor 47/48) vom Basic-Interpreter vorgenommen. Sind aber vorher schon Arrays in die Array-Liste (ab Vektor 47/48) eingetragen worden, dann muß zuerst der ganze Array-Listen-Block um 7 Byte nach oben verschoben werden, um für diese neue Variable N den nötigen Platz zu schaffen. Dies geschieht für jede neue Variable. Sie können sich vorstellen, daß solch ein Weg nicht gerade zur Beschleunigung eines Programmablaufes beiträgt. Man sollte also einfach in den ersten Zeilen eines Programmes schon alle Variablen definieren. Wenn man von einigen noch keine Werte angeben kann,

weil die veränderlich sind, setzt man sogenannte Dummys ein, also irgendwelche erfundenen Beträge.

Die Hauptsache ist ja nur die Vorbelegung aller Listenplätze der Variablenliste. Dabei kann man dem Computer gleich noch etwas Gutes tun, indem man die am häufigsten gebrauchten Variablen ganz an den Anfang setzt. Der Interpreter muß dann nur immer kurze Teile der Liste beim Aufsuchen durchforsten.

Die Arrays, deren Listenbeginn durch den zuletzt betrachteten Vektor 47/48 angezeigt werden, sollen zu einem späteren Zeitpunkt ausführlich behandelt werden. Hier deshalb nur kurz die Übersicht über die drei Typen, die wir im Commodore 64 auswählen können:

Gleitkomma-Arrays	zum Beispiel A(N)
Integer-Arrays	zum Beispiel B%(N)
String-Arrays	zum Beispiel C$(N)

Jeder dieser Typen kann in unterschiedlicher Dimensionierung auftreten:

eindimensionales Array	zum Beispiel A(N) (Bild 5)
zweidimensionales Array	zum Beispiel B(N,M) (Bild 6)
dreidimensionales Array	zum Beispiel C(N,M,O) (Bild 7)

und so weiter.

Die dritte Dimension

Theoretisch können im Commodore 64 sogar Arrays mit 256 Dimensionen definiert werden, solange der verfügbare Speicherplatz ausreicht. Einen Zeiger auf das Ende der Array-Liste finden wir in 49/50. Wenn Sie wieder in unserer Direkt-Eingabe das Ergebnisfeld löschen und dann für A den Wert 49 einsetzen, dann zeigt der Bildschirm wie vorhin:
»10 8 2058«.

Weil wir kein Array verwendet haben, ist der Array-Ende-Zeiger mit dem Array-Beginn-Zeiger identisch. Etwas Neues erfahren wir, wenn wir für A nun mal 51 einsetzen. Es ergibt sich der Ausdruck:
»0 160 40960«.

Diese Zahl kennen wir als die Startadresse des Basic-Interpreter-ROMs. Gleichzeitig haben wir hier das Ende des normalerweise verfügbaren Basic-Speicherraumes vor uns. 51/52 ist aber nicht etwa der Zeiger auf das Ende des Basic-Speichers. Vielmehr gibt dieser Vektor Auskunft über den Anfang des Stringspeichers. Das kann man leicht feststellen. Geben Sie doch mal ein: »A$ = "123456"«.

Dann löschen Sie das Ergebnisfeld unserer Direktzeile und fragen Sie erneut nach dem Inhalt des Vektors 51/52 durch Hochfahren des Cursors zur Direktzeile und Drücken von RETURN. Auf dem Bildschirm erscheint nun:
»250 159 40954«.

Der Vektor ist jetzt sechs Speicherplätze weiter nach unten gerutscht, also für jedes Zeichen unseres Teststrings A$ um einen Speicherplatz. Wir sehen daran, daß Strings am oberen Ende unseres Basic-Speichers abgelegt und neue Strings von da an abwärts angehängt werden. 51/52 weist daher auf den jeweils aktuellen unteren Rand des Stringspeichers. Zwischen der durch 51/52 angezeigten Speicherposition und der durch 49/50 definierten Array-Obergrenze liegt freier Speicherraum. Beide Zeiger wandern im Verlauf eines Programmes mit vielen Variablen aufeinander zu. Wenn ihr Abstand eine gewisse Mindestgrenze erreicht, tritt die sogenannte Garbage-collection (zu deutsch Müll-Sammlung) ein. Nicht mehr benötigte Strings werden dabei (etwas vereinfacht) gelöscht und noch gültige soweit wie möglich nach oben gerückt. Dann klafft wieder freier Speicher zwischen String- und Array-Tabelle. Sollte aber irgendwann auch die Garbage-collection keinen freien Speicher mehr schaffen

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}

Bild 5. Eindimensionales Array

A_{00}	A_{01}	A_{02}	A_{03}	A_{04}	A_{05}	A_{06}
A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}
A_{20}	A_{21}	A_{22}	A_{23}	A_{24}	A_{25}	A_{26}
A_{30}	A_{31}	A_{32}	A_{33}	A_{34}	A_{35}	A_{36}
A_{40}	A_{41}	A_{42}	A_{43}	A_{44}	A_{45}	A_{46}
A_{50}	A_{51}	A_{52}	A_{53}	A_{54}	A_{55}	A_{56}
A_{60}	A_{61}	A_{62}	A_{63}	A_{64}	A_{65}	A_{66}

Bild 6. Zweidimensionales Array

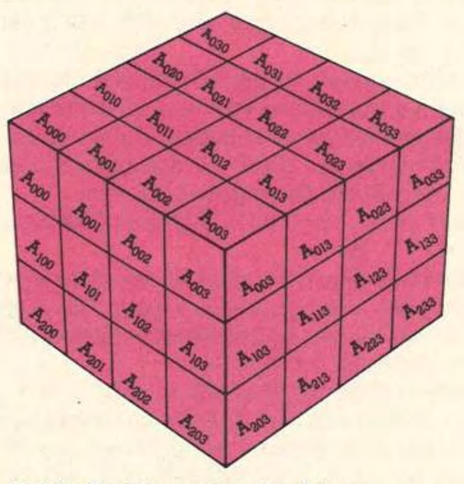


Bild 7. Dreidimensionales Array

Tabelle 2. Zeropage-Vektoren, die ein Basic-Programm organisieren

Commodore-Name	Adresse LSB	Adresse MSB	Normaler Inhalt LSB	Normaler Inhalt MSB	Dezimal	Bedeutung
TXTTAB	43	44	1	8	2049	Basic-Text-Anfang
VARTAB	45	46	3	8	2051	Variablen-Tabelle Anfang
ARYTAB	47	48	3	8	2051	Array-Tabelle Anfang
STREND	49	50	3	8	2051	Array-Tabelle Ende
FRETOP	51	52	0	160	40960	String-Tabelle Spitze
MEMSIZ	55	56	0	160	40960	Basic-Speicher-Ende

können, dann meldet der Computer einen »OUT OF MEMORY ERROR«.

Der Vektor 53/54 soll uns nicht belasten. Es handelt sich um einen Hilfszeiger zur Stringverarbeitung. Viel interessanter sind die Speicherplätze 55 und 56 für uns, die nun wirklich das Ende des verfügbaren Basic-Speichers enthalten. Steht die Direktzeile noch auf dem Bildschirm? Dann ersetzen Sie bitte den A-Wert durch 55 und – nach vorherigem Löschen des Ergebnisfeldes sowie einem RETURN in der Direktzeile – sehen Sie den Ausdruck:

»0 160 40960«.

Insgesamt sehen Sie in Tabelle 2 eine Zusammenfassung der Namen, Speichernummern und Inhalte all dieser Vektoren im Einschaltzustand unseres Commodore 64. Bild 8 zeigt Ihnen den Basic-Speicher mit den Zeigern. Hier soll ein Programm im Speicher liegen. Wir können aber noch mehr tun, als nur diese Vektoren mit PEEK lesen. Hier liegt ja RAM vor, das durch POKE-Kommandos veränderbar ist.

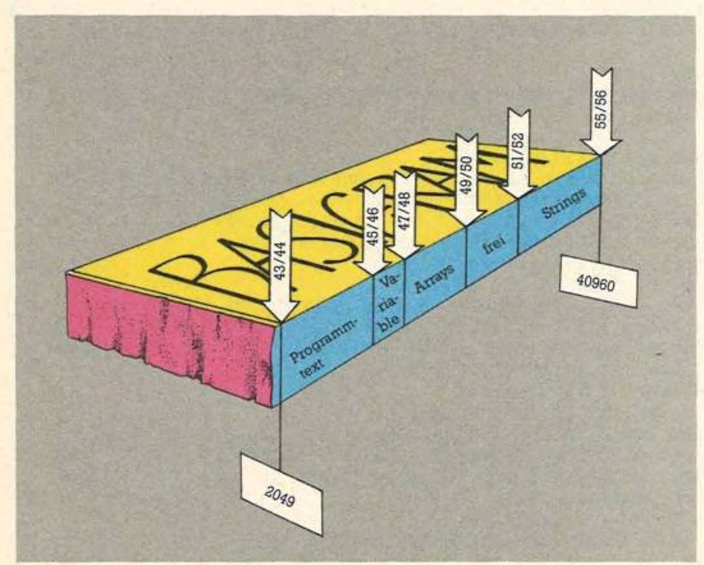


Bild 8. Das normale Basic-RAM mit den Vektoren

Veränderungen am Basic-Speicher

Folgendes kann man am Basic-Speicher alles ändern:

1) Herabsetzen des Basic-Speicherendes
2) Heraufsetzen des Basic-Speicherbeginns
3) Mehrere Basic-Programme im Speicher gleichzeitig list- und lauffähig aufbewahren.

Die am häufigsten gestellte Aufgabe ist das Schützen des oberen Basic-Speicherbereichs vor dem Überschreiben durch Basic. Zwar bietet der Commodore 64 ab Speicherstelle 49152 (siehe Bild 2) einen 4 KByte großen Speicherraum, der für Maschinencode-Programme oder ähnliches recht gut geeignet ist, trotzdem muß man manchmal solche Programme auch in den Basic-Speicher legen. Beispielsweise fangen Modulprogramme meistens bei der Speicherstelle 32768 an, was mit einem besonderen Verhalten unseres Computers beim Einschalten, aber auch beim Reset, zusammenhängt.

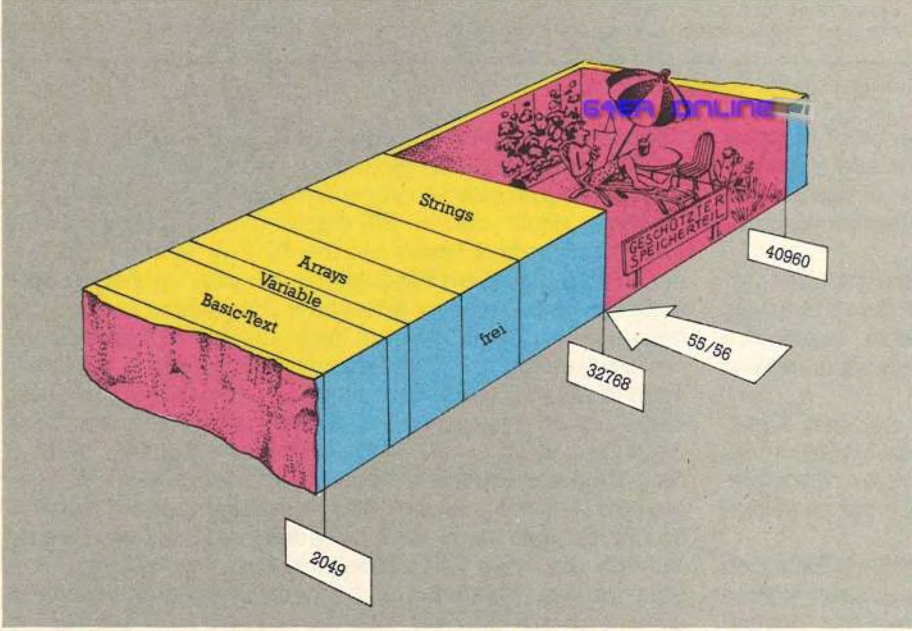


Bild 9. Oberhalb von Adresse 32768 (Vektor 55/56) ist das Basic-RAM geschützt

Nehmen wir also an, wir möchten den Speicherraum ab 32768 schützen. Dann können wir uns zunächst der inzwischen schon bekannten Formeln bedienen:

MSB = INT(32768/256) = 128
LSB = 32768 - 256*128 = 0

Diese Werte müssen wir noch in den Vektor 55/56 schreiben. Normalerweise wäre noch eine Korrektur des Stringvektors notwendig, doch der NEW-Befehl erledigt das für uns. Also:

```
POKE 55,0:POKE 56,128:NEW
```

Der POKE 55,0 kann auch weggelassen werden, da der LSB des Vektors meist sowieso auf Null steht.

Prinzipiell gibt es zwei Möglichkeiten, die sich durch die Behandlung der Variablen unterscheiden (siehe dazu Bild 10 und Bild 11): Bei der ersten Variante werden die Programme relativ dicht hintereinander gelegt. Oberhalb des letzten Programms ist dann ein gemeinsamer Variablen-, Array- und

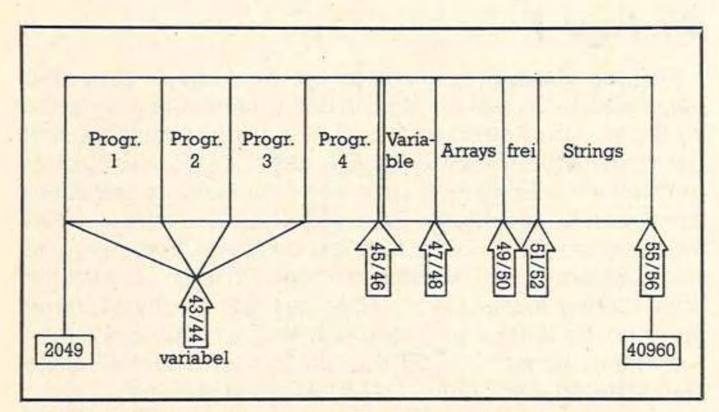


Bild 10. Basic-Programme und Variablen hintereinander

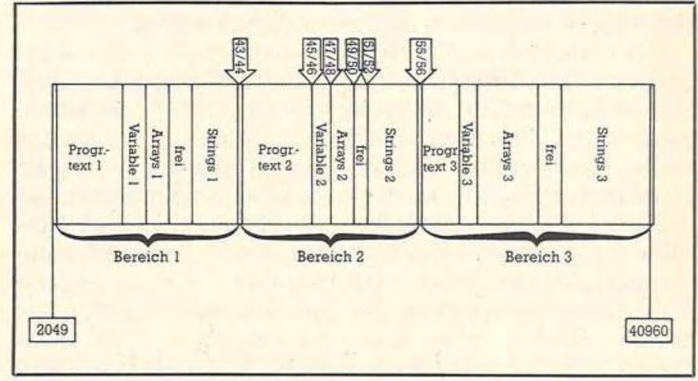


Bild 11. Programme und Variablen gemeinsam im Speicher

Stringspeicher vorhanden. Das erreicht man so:

1) Einladen des ersten Programms und über den Vektor 45/46 feststellen, wo der Programmtext endet.
2) Die nächste volle Pageadresse wählen (diese müssen Sie sich unbedingt aufschreiben, sonst finden Sie Ihr Programm später nicht mehr) und – wie bereits beschrieben – eine Null einPOKEn, sowie den Basic-Anfang-Vektor auf diese Adresse +1 setzen.
3) Nächstes Programm laden und wieder wie in 1) und 2) geschildert verfahren, bis alle benötigten Programme im Speicher liegen.
4) Automatisch weist nun der Programmende-Vektor 45/46 auf das Ende des letzten Programms. Von dort an werden dann die Variablen und Arrays abgelegt.
5) Weil Sie glatte Pages als Startadressen gewählt haben, genügt es, zum Umschalten den jeweils dazugehörigen Wert aus Punkt 2 in Speicherstelle 44 zu POKEn.

Die Vorteile dieses Verfahrens liegen in der einfachen Umschaltung und der Tatsache, daß man so ziemlich viele Programme im Speicher halten kann. Von Nachteil ist es, daß nachträglich Änderungen eines Programmes sehr vorsichtig vorgenommen werden müssen, damit nicht das nachfolgende Programm überschrieben wird. Auch kann man nicht einzelne Programme speichern, sondern immer nur alle zugleich. Außerdem gibt es manchmal Situationen, in denen die gemeinsame Nutzung eines Variablenspeichers von Nachteil ist.

```
1 REM *********************************
2 REM *                               *
3 REM *  PROGRAMM 1                   *
4 REM *                               *
5 REM *  C.SAUERS BASIC-SWITCH FUER   *
6 REM *  DEN C-64 UMGESCHRIEBEN VON   *
7 REM *  H. PONNATH HAMBURG 1985      *
8 REM *                               *
9 REM *********************************
10 REM
15 S=0:M=0:A=0:P=49152:K=0
20 READA:PRINTCHR$(147)A:IFA=-1THEN50
25 POKEP,A
30 S=S+A:P=P+1:GOTO20
50 READK:M=M+1:IFS<>KTHENPRINT"DATA-FEHLER IN BLOCK "M:END
55 S=0:IFM=4THENCLR:END
60 GOTO20
99 REM ******* DATA-BLOCK 1 ***********
100 DATA230,122,208,2,230,123,32,121,0,201,255,240,3,76,121,0,32,115,0,201
101 DATA69,208,3,76,168,192,201,83,240,3,76,8,175,32,155,183,165,101,240
102 DATA4,228,251,144,3,76,72,178,134,252,32,133,192,166,252,202.138,10,10
103 DATA10,168,162,0,185,23,193,149,43,200,232,224,-1,8756
109 REM ****** DATA-BLOCK 2 ***********
110 DATA4,208,245,162,0,185,23
111 DATA193,149,55,200,232,224,2,208,245,165,252,133,250,32,96,166,32,142
112 DATA166,32,115,0,240,6,201,143,240,16,208,245,160,1,177,122,208,239,200
113 DATA192,2,208,234,76,116,164,32,215,170,165,122,164,123,32,30,171,240
114 DATA241,166,250,202,138,10,10,10,-1,9805
119 REM ****** DATA-BLOCK 3 ***********
120 DATA168,162,0,181,43,153,23,193,200,232
121 DATA224,4,208,245,162,0,181,55,153,23,193,200,232,224,2,208,245,96,32
122 DATA133,192,32,115,0,32,138,173,32,191,177,165,253,133,43,165,254,133
123 DATA44,165,101,24,133,253,165,100,101,254,133,254,165,251,133,252,230
124 DATA251,32,22,228,165,254,-1,10113
129 REM ****** DATA-BLOCK 4 ***********
130 DATA205,132,2,144,16,208,7,165,253,205,131,2,144
131 DATA7,169,12,160,173,76,30,171,165,253,133,55,165,254,133,56,32,68,166
132 DATA165,252,133,250,76,49,192,162,0,134,250,142,116,0,232,134,251,134
133 DATA252,165,43,133,253,165,44,133,254,169,76,133,115,169,192,133,117
134 DATA96,255,0,-1,9461
135 REM ****** ENDE DER DATAS *********
```

Listing 4. Basic-Switch für C64

Die zweite Variante weist diese Mankos nicht auf, dafür verbraucht sie aber mehr Speicherplatz. Außerdem erfolgt die Umschaltung von Programm zu Programm wesentlich umständlicher. Hier müssen nämlich immer alle Vektoren von 43/44 bis 55/56 umgestellt werden.

Das Prinzip dieser zweiten Variante ist es, den Basic-Speicherraum jedes Programms mit seinen Variablen als allein vorhandenen Basic-Speicher zu definieren. Der Computer nimmt den gesamten anderen Bereich mit all seinen Inhalten nicht mehr wahr.

Hier nun die Erklärung des Listings 4: Nach dem Eintippen speichern Sie bitte das Programm erst einmal. Dann starten Sie es mit »RUN«. Der Basic-Lader schreibt nun das Maschinenprogramm aus den DATA-Zeilen in den Speicherbereich ab 49152, wo es gegen den Basic-Zugriff geschützt liegt. Nach der READY-Meldung können Sie durch »NEW« den Basic-Lader löschen; er wird nicht mehr benötigt.

Das Maschinencode-Programm wird durch »SYS 49401« gestartet. Davon merken Sie aber zunächst überhaupt nichts. Erst dann, wenn Sie im Direktmodus mal ein Pi eingeben, gefolgt von E oder S, wird das Programm tätig. Bevor Sie das tun, müssen Sie aber noch ein wenig Kopfarbeit leisten. Es ist nämlich nötig, zu überlegen, wieviel Speicherraum ein Programm inklusive seiner Variablen beanspruchen wird. Da sollte man nicht zu kleinlich sein, denn eine nachträgliche Änderung ist nicht mehr möglich. Wenn Sie diese Angabe parat haben, dann richten Sie zum Beispiel mit dem Kommando πE3000 einen Speicherbereich von 3000 Byte (minus 2) für Ihr erstes Programm ein. Diesem Bereich wird die laufende Nummer 1 zugeordnet. Mit weiteren πE-Kommandos können Sie nun – solange der Speichervorrat reicht – die Speicherbereiche 2, 3, und so weiter einrichten. Wenn Sie nun in einem bestimmten Speicherbereich arbeiten möchten, dann schalten Sie diesen ein, beispielsweise mit »πS1« den ersten Speicherbereich. Hier können Sie nun nach Herzenslust ein Programm laden oder schreiben oder auch nur eine Directory lesen oder Programme ändern (Sie müsen nur innerhalb des von Ihnen beim Einrichten gesetzten Limits bleiben). Soll es ein anderer Speicherbereich sein? Einfach mit »πS (Bereichsnummer)« umschalten! Ein gutes Mittel zur Orientierung hat C. Sauer in sein Programm noch eingebaut, welches ich mit übernommen habe. Die Orientierung kann nämlich spätestens nach Laden des dritten Programms schwer werden! Das Maschinencode-Programm druckt deshalb nach dem Umschalten in einen anderen Bereich die erste REM-Zeile des dort vorhandenen Programms aus. Man kann sich daher Orientierungshilfen in einer solchen REM-Zeile notieren, was sehr zu empfehlen ist!

Die Page 1

Für den Basic-Programmierer ist die Page 1 ganz flott abgehandelt: Da gibt es nämlich nichts Interessantes für ihn zu finden. Von Speicheradresse 256 bis 511 befindet sich der sogenannte Prozessorstack, der auch Stapelspeicher genannt wird. Ein Eingriff kann hier zu besonders eleganten Abstürzen führen. Dieser Prozessorstack und sein Inhalt ist eines der empfindlichsten Gebiete unseres Computers und nur sehr erfahrene Assembler-Programmierer können ihn ohne Scheu manipulieren. Aber das soll Sie nicht davon abhalten, Ihr Glück auch hier durch ein paar »POKEs« oder – wesentlich harmloser – PEEKs zu versuchen. Die Notbremse (Ausschalten des Computers) funktioniert ja immer.

(Heino Ponnath/tr)



Monitor oder Fernseher?

Es läßt sich viel Geld sparen, wenn der vorhandene Fernseher am Computer angeschlossen wird. Doch die meisten Computerbesitzer entschließen sich nach einiger Zeit doch zum Kauf eines zweiten Bildschirmgerätes. Aber soll das nun ein Monitor sein, oder ist ein tragbarer Fernseher zweckmäßiger?

Bildschirmgeräte stehen heute in fast allen Haushalten – in Form eines Fernsehgerätes. Was liegt also für den sparsamen Computerkäufer näher als diese Tatsache zu nutzen und vorläufig auf den Kauf eines Monitors zu verzichten. Nach einigen Tagen treten jedoch meist Probleme auf. Einerseits wollen andere Familienmitglieder lieber Professor Brinkmann sehen als das neueste Actionspiel, andererseits wirkt der große Fernsehbildschirm auf die Dauer ermüdend für die Augen. Und die Bildqualität ist auch nicht gerade die beste – aber warum?

Verdeutlichen wir uns dazu, wie der Computer ein Bild auf den Fernseher bringt. Dem Computer steht der Bildinhalt in Form digitaler Daten zur Verfügung. Diese werden zu einem Videosignal aufbereitet. Das Videosignal wird wiederum zu einem hochfrequenten Antennensignal umgeformt und über die Antennenbuchse dem Fernseher zugeführt. Im Fernsehgerät muß auf komplizierte Weise wieder ein Videosignal zurückgewonnen werden. Dadurch kann sich die Qualität des ursprünglichen Signals erheblich verschlechtern – je nach Qualität des Fernsehgerätes mehr oder weniger.

Der Wunsch entsteht, auch einmal vor einem scharfen Bildschirm zu sitzen. Ein zweites Gerät muß also her. Ein tragbarer Fernseher mit kleinem Bildschirm und Video-Eingang bringt gegenüber dem oben beschriebenen Beispiel schon eine deutliche Verbesserung. Aber auch hier gibt es erhebliche Qualitätsunterschiede.

Probleme bei der Textwiedergabe auf dem Bildschirm

In den Bildern 1 und 2 sehen Sie Text- (40-Zeichen-Darstellung) und Grafikausschnitte von tragbaren Fernsehgeräten mittlerer und guter Qualität. Wie Sie aus den Bildern erkennen können, ist es besonders schwierig, bei der Textwiedergabe scharfe Kanten auf dem Bildschirm zu erzeugen. Woran liegt das? Nun, die Bandbreite des Videoverstärkers im Fernseher ist dafür verantwortlich. Wir wollen Ihnen auch den Zusammenhang erklären. In einem Fernsehgerät werden 15625 Bildzeilen pro Sekunde geschrieben. Nehmen wir an, es sollen 100 senkrechte Striche auf den Bildschirm geschrieben werden. Die Striche sind natürlich auch in jeder einzelnen Zeile enthalten. Dadurch erhöht sich die Frequenz nochmals um den Faktor 100 auf 1 562 500 Hz, also zirka 1,5 MHz. Die meisten Fernsehgeräte haben eine Video-Bandbreite von 5 bis 7 MHz, in Ausnahmen bis 10 MHz. Also – kein Problem mit den 100 Strichen – sollte man meinen. Dem ist aber anders. Wenn Striche oder Buchstaben auf dem Bildschirm dargestellt sind, sollen sie sich auch scharf vom Hintergrund abgrenzen. Für das Videosignal bedeutet das einen rechteckigen Spannungsverlauf. Und da liegt auch der Hase im Pfeffer. Denn ein Rechtecksignal setzt



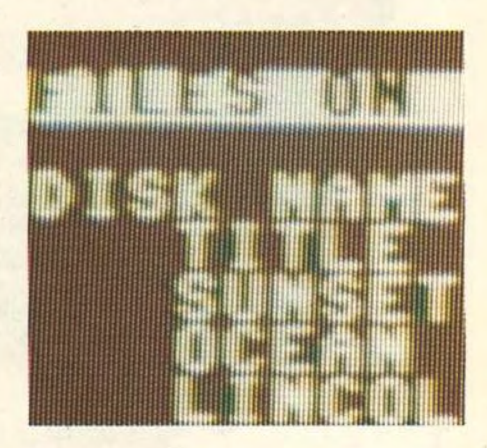
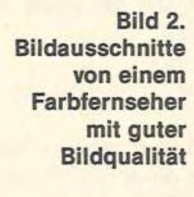


Bild 1. Bildausschnitte von einem Farbfernseher mit mittlerer Bildqualität



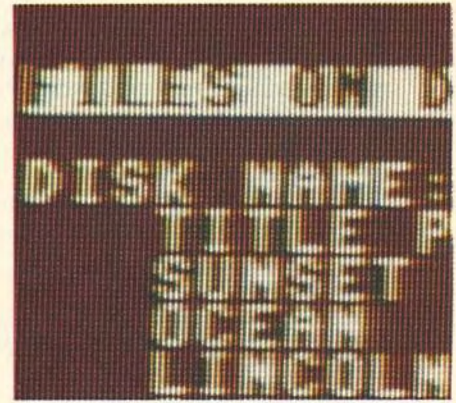
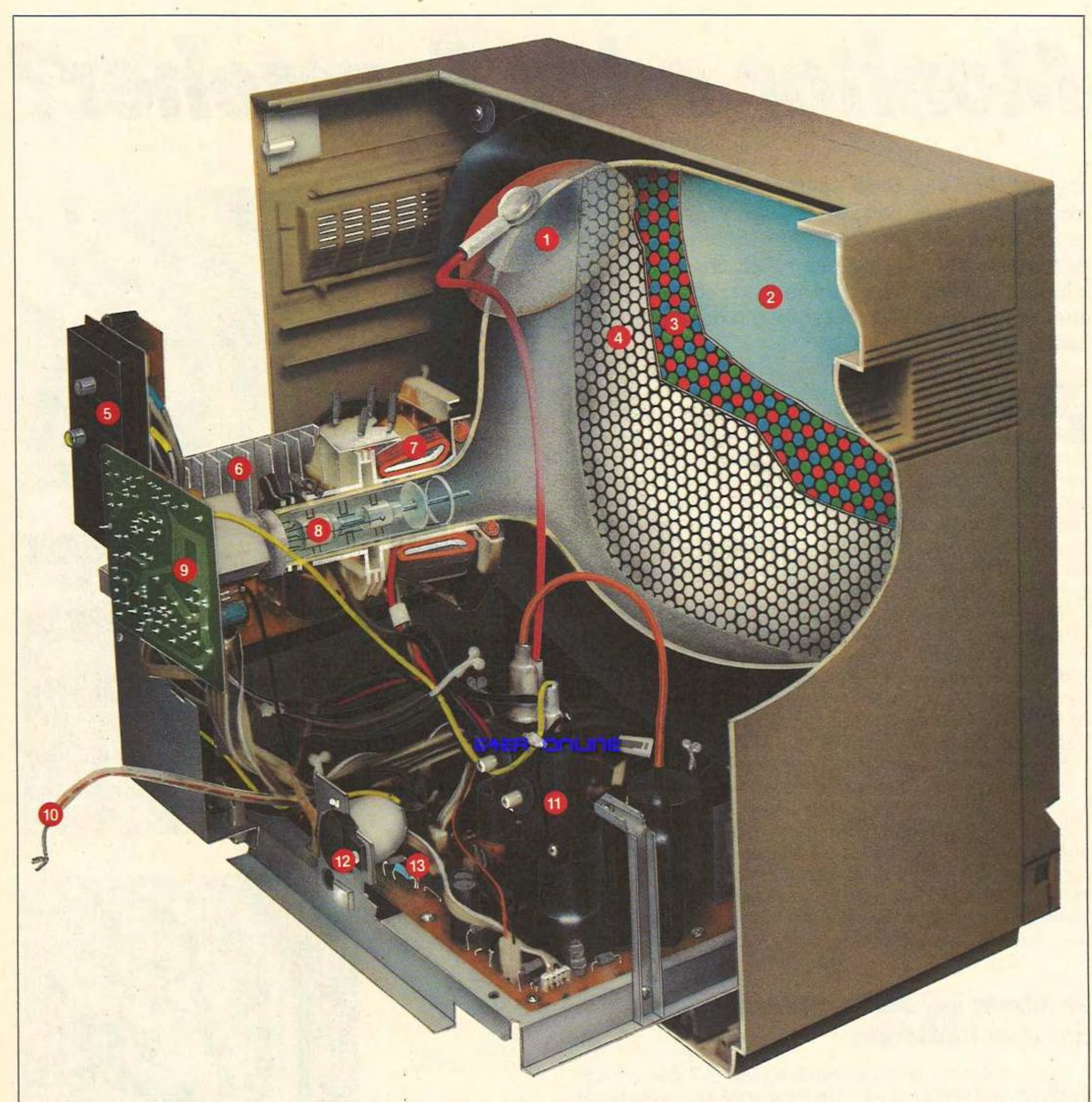


Bild 2. Bildausschnitte von einem Farbfernseher mit guter Bildqualität



Einblick in das Innere eines Monitors

1 Hochspannungsanschluß der Bildröhre
2 Glaskolben der Bildröhre (verstärkte Frontscheibe)
3 Farbtripel
Zu einem Farbtripel gehört je ein rotes, grünes und blaues Feld. Die Anzahl der Farbtripel einer Bildröhre ist auch ein Kriterium für die mögliche Bildauflösung. Je mehr Tripel vorhanden sind, um so höher ist die Bildauflösung.
4 Lochmaske
Zu jedem Farbtripel gehört ein Loch der Lochmaske.
5 Anschlußplatte für die Eingangssignale
6 Netzteil mit Kühlrippen
7 Ablenkeinheit
Die Elektronenstrahlen werden durch den Strom, der im jeweiligen Moment durch die Ablenkspulen fließt, in eine bestimmte Richtung gelenkt. Auf diese Weise können die Elektronenstrahlen von oben nach unten Zeile für Zeile über den Bildschirm gelenkt werden.
8 Elektronenkanone
Hier werden drei Elektronenstrahlen erzeugt. Je einer für Rot, Grün und Blau.
9 Bildröhrenanschluß-Platine
10 Lautsprecherkabel
Es liegt hier frei, da sich der Lautsprecher in der abgenommenen Rückwand befindet.
11 Hochspannungsteil
Die Hochspannung für die Bildröhre wird hier erzeugt.
12 Umschalter für die Netzspannung (110/220 Volt)
13 Videoverstärker und weitere Elektronik (kn)

sich mathematisch zusammen aus der sinusförmigen Grundfrequenz und den ungeraden Vielfachen dieser Frequenz. Für unser Beispiel bedeutet dies eine Zusammensetzung aus 1,5 MHz und 4,5 MHz und 7,5 MHz und 10,5 MHz und so weiter. Vielleicht können Sie sich jetzt schon denken, warum eine große Bandbreite so wichtig ist. Schauen wir uns aber ein Beispiel an.

So werden die Kanten deutlich

Im Bild 3 sehen Sie ein Rechteck-Signal, das sich aus Grundschwingung und erster Oberschwingung zusammensetzt. Für unser Beispiel entspräche das einer Bandbreite von 4,5 MHz. Wie Sie sehen, ist es gar nicht möglich, bei 100 senkrechten Strichen scharfe Kanten auf dem Bildschirm wiederzugeben. Mit drei Oberschwingungen (Bild 4) kommen wir schon an ein Rechteck-Signal heran. Eine Bandbreite von 10,5 MHz ist dafür bei unserem Beispiel notwendig. Mit einem sehr guten Fernsehgerät ist dies möglich. Mit sechs Oberschwingungen haben wir schon ein nahezu perfektes Rechteck-Signal (Bild 5). Allerdings ist bei 100 senkrechten Strichen dafür schon eine Bandbreite von 19,5 MHz notwendig. Eine solche Bandbreite bieten jedoch nur Monitore. Im Bild 6 sehen Sie die bereits bekannten Bildausschnitte, aufgenommen von dem Monitor Commodore 1701 - eine deutliche Verbesserung gegenüber den Fernsehgeräten.

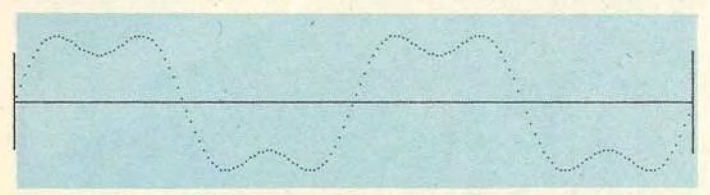


Bild 3. Rechteck-Signal aus Grundschwingung und einer Oberschwingung

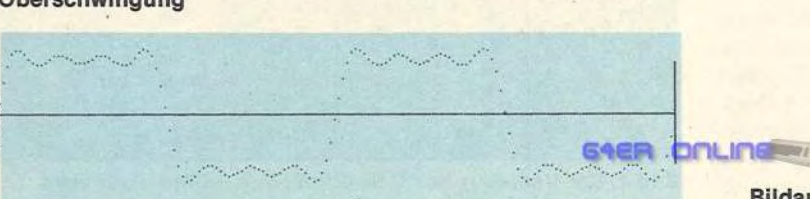


Bild 4. Rechteck-Signal aus Grundschwingung und drei Oberschwingungen

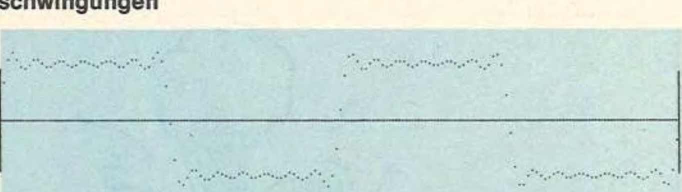


Bild 5. Rechteck-Signal aus Grundschwingung und sechs Oberschwingungen

Wenn Sie sich zum Kauf eines neuen Bildschirmgerätes entschließen, so überlegen Sie sich genau, wofür Sie ihren Computer verwenden wollen. Falls Sie ausschließlich an Spielen interessiert sind, so ist ein tragbarer Farbfernseher die beste Lösung. Die Bildschärfe reicht für Spiele aus, wenn ein Videoeingang vorhanden ist, und Sie können das Gerät zusätzlich als normalen Fernseher verwenden. Wenn Sie aber ein Grafikfan sind, dann ist ein Farbmonitor das richtige Gerät für Sie.

Anders ist es, wenn Sie hauptsächlich Textverarbeitung oder Datenverwaltung machen wollen, wobei Sie auf Farbe vollkommen verzichten können. Eine dritte Möglichkeit bietet sich hier an, der Kauf eines monochromen Monitors. Erstens schonen Sie damit Ihre Brieftasche, denn monochrome Monitore sind schon für 300 bis 400 Mark zu haben. Zweitens schonen Sie Ihre Augen. Die Textwiedergabe auf dem Bildschirm ist deutlich schärfer als bei einem Farbmonitor. In den Bildern 7 und 8 können Sie die Textwiedergabe bei 80 Zeichen pro Zeile zwischen einem Farbmonitor und einem monochromen Monitor vergleichen. Falls Sie einen Lichtgriffel verwenden wollen, so sollten Sie sich keinen bernsteinfarbenen Monitor zulegen, da diese nachleuchtende Bildröhren besitzen. Dem Computer werden dadurch falsche Positionsdaten übergeben. Daß viele monochrome Monitore keinen Lautsprecher haben, sollten Sie beim Kauf ebenfalls berücksichtigen.

Wenn Sie sich für einen Monitor entschlossen haben, sollten Sie unbedingt prüfen, ob Sie Ihren Computer problemlos daran anschließen können.

Insgesamt scheint allerdings ein tragbarer Farbfernseher mit Videoeingang für den Anfang ein guter Kompromiß zu sein. Achten Sie aber darauf, daß die Bildschirmgröße nicht über 40 cm Bildschirmdiagonale liegt. Sonst wird das Arbeiten in einem Abstand von zirka einem Meter vor dem Bildschirm schnell anstrengend. (kn)





Bild 6. Bildausschnitte von einem Monitor (Commodore 1701)



Bild 7. Text mit einer 80-Zeichen-Karte beim Farbmonitor (Commodore 1701)

Bild 8. Text mit einer 80-Zeichen-Karte auf einem monochromen Monitor (Zenith)



Die Ports

Der Expansion-Port und der User-Port sind zwei Anschlüsse, die Ihrem Computer Tür und Tor für

Wollen Sie Programm-Module an Ihrem Computer betreiben, einen Akustikkoppler anschließen oder eine Alarmanlage überwachen? Dann müssen Sie den User- oder den Expansion-Port benutzen. Natürlich sind dies nur drei Beispiele unter einer Vielzahl von Möglichkeiten. Es sind aber auch Beipiele, die typisch für die Verwendung des jeweiligen Ports sind. So schließt man Akustikkoppler und Alarmanlage am User-Port (Bild 1) an, während in den Expansion-Port (Bild 2) Programm-Module gesteckt werden. Aber wenden wir uns erst einmal dem User-Port zu.

Der vielseitige Port

Der User-Port ist eine Schnittstelle, die Sie sehr vielseitig verwenden können. Eine für die meisten Akustikkoppler notwendige RS232-Schnittstelle läßt sich über den User-Port ebenso realisieren wie Meß- und Steueraufgaben.

Sehen wir uns diese Schnittstelle etwas genauer an. Die Anschlußbelegung des User-Ports zeigt das Bild 3. Beachten Sie bitte, daß die Anschlüsse an der Ober- und Unterseite des User-Ports verschiedene Funktionen haben. Klemmen Sie deshalb niemals eine Krokodilklemme oder etwas Vergleichbares an!

Wie schon gesagt, ist der User-Port sehr variabel verwendbar. Der Grund dafür ist ein im User-Port enthaltener 8-Bit-Parallelport (Bild 3: PB0 bis PB7), den Sie frei programmieren können. Das bedeutet, über den Parallel-Port können Sie Daten einlesen und ausgeben. Interessant dabei ist, daß Sie die Datenrichtung für jede der acht Leitungen einzeln bestimmen können. Welche Leitung als Eingang und welche als Ausgang geschaltet wird, geben Sie über das Datenrichtungsregister vor. Die Abkürzung »DDR« für dieses Register kommt aus dem Englischen und heißt »Data Direction Register«. Aber wie können mit einem Register acht Leitungen gesteuert werden? Nun, dafür müssen wir uns etwas mit dem Zusammenhang zwischen Dual- und Dezimalzahlen beschäftigen.

Der User-Port wird programmiert

Das duale Zahlensystem, mit dem der Computer arbeitet, kennt nur zwei Ziffern, die 0 und die 1. Dabei wird eine Informationseinheit, die 0 oder 1 enthält, ein »Bit« genannt. Es ist die kleinste, in der Cumputertechnik verwendete Informationseinheit. Mit einem Bit lassen sich die Dezimalzahlen 0 ($=0 * 2^0$) und 1 ($=1 * 2^0$) darstellen. Nehmen wir ein zweites Bit dazu, so können wir schon Dezimalzahlen von 0 bis 3 darstellen. Eine Drei ergibt sich, wenn beide Bits eine 1 enthalten ($1 * 2^1 + 1 * 2^0 = 2 + 1 = 3$). Mit acht Bits können schon Dezimalzahlen von 0 bis 255 dargestellt werden. Im Bild 4 finden Sie als Beispiel, wie die Zahl 223 im dualen System zerlegt ist. Acht Bits ergeben übrigens in der Computertechnik die nächstgrößere Informationseinheit, ein »Byte«. Ihr C64 hat einen Speicherbereich von 65536 Byte. Das Byte 56579 ist das oben erwähnte Datenrichtungsregister. Wenn Sie dort eine Eins im Bit 0 eingeben, wird die dazugehörige Leitung PB0 auf Ausgang geschaltet. Überprüfen Sie doch einmal das DDR nach den Einschalten, indem Sie eingeben:

```
PRINT PEEK (56579)
```

Auf dem Bildschirm müßte anschließend eine 0 erscheinen, da nach dem Einschalten des Computers die Portleitungen PB0 bis PB7 als Eingänge geschaltet sind. Wenn Sie den Wert aus unserem Beispiel (Bild 4) mit

```
POKE 56579,223
```

eingeben, so enthält das DDR eine Bitfolge von 11011111. Alle PB-Leitungen sind damit als Ausgänge geschaltet, bis auf die Leitung PB5.

Über das DPB-Register (Data Port B) in der Speicherzelle 56577 können Sie nun die einzelnen Ausgänge auf 0 oder

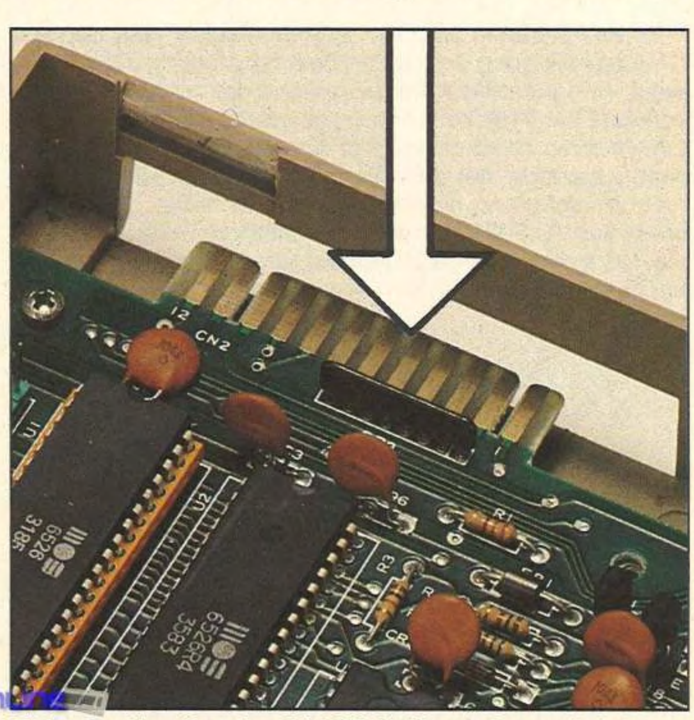
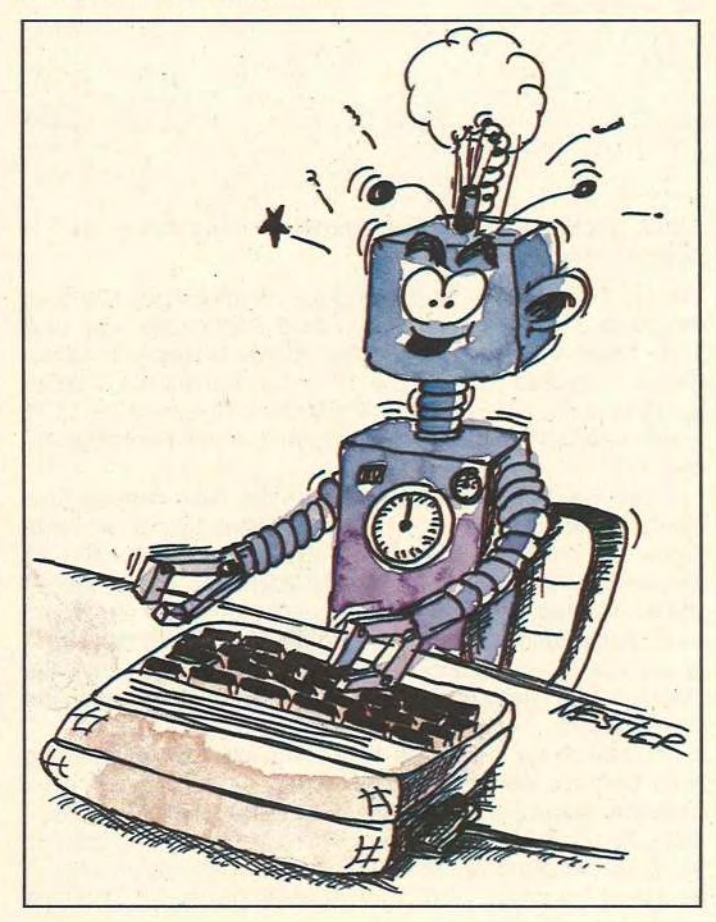


Bild 1. Der User-Port des C64 (Pfeil). Von rechts nach links sehen Sie die Kontakte 1 bis 12. Die Kontakte A bis N sind auf der unteren Platinenseite.



des C64

die verschiedensten Erweiterungen öffnen. Was können Sie mit den einzelnen Ports machen?

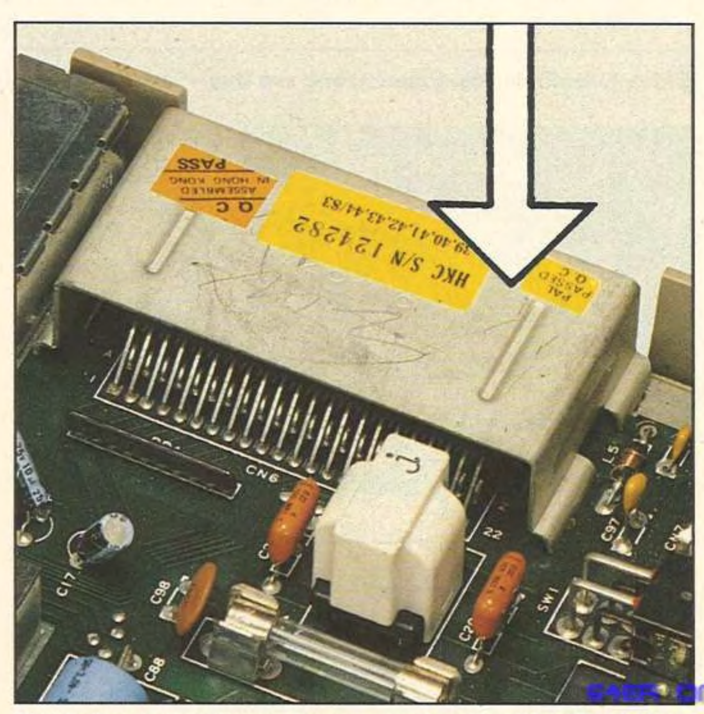


Bild 2. Der abgeschirmte Expansion-Port des C64 (Pfeil). Die 44 Kontakte sind über Drähte an der Platine angeschlossen.

auf 1 setzen - in gleicher Weise, wie Sie es mit den DDR getan haben. Mit

```
POKE 56577,222
```

haben Sie dort eine Bitfolge von 11011110 eingegeben. Bis auf PB0 sind so alle Ausgänge auf 1 gesetzt. Das bedeutet, an den Anschluß-Pins D bis H (PB1-PB4), sowie an Pin K (PB6) und L (PB7) liegen +5 Volt, während Pin C (PB0) auf 0 Volt liegt. Im vorliegenden Fall wurde im Bit 5 eine 0 eingegeben. Da die entsprechende Leitung im DDR jedoch vorher als Eingang geschaltet wurde, ignoriert der Computer das Bit 5 des DPB-Registers. Sie hätten ebensogut mit

```
POKE 56577,254
```

eine Bitfolge von 11111110 eingeben können.

Das müssen Sie beim Programmieren des User-Ports beachten

Wenn Sie nun anfangen, mit dem User-Port zu experimentieren, müssen Sie allerdings etwas aufpassen. Denn wie wir schon erwähnt haben, werden die auf 1 gesetzten Ausgänge von einem integrierten Baustein, dem CIA (Control Interface Adapter), auf +5 Volt geschaltet. Verbinden Sie diese Anschlüsse nun aus Versehen mit Masse, so nimmt Ihnen der CIA diese Behandlung sehr übel. Anschließend müssen Sie höchstwahrscheinlich Ihren Computer in eine Service-Werkstatt geben. Also, seien Sie bitte vorsichtig.

Wenn Sie den Port als Eingang benutzen, ist es natürlich auch interessant, einzelne Leitungen zu überprüfen. Mit Hilfe des Port-Registers können Sie dieses tun. Nach

```
PRINT PEEK (56577)
```

erscheint der Wert des DPB-Registers auf dem Bildschirm. Schalten Sie Ihren Computer noch einmal an, und überprüfen Sie das DPB-Register. Sie werden feststellen, daß alle Bits auf 1 gesetzt sind. Alle Leitungen des Port B liegen demnach auf +5 Volt.

Mit dem User-Port prüfen

Wie Sie sich sicher erinnern werden, sind nach dem Einschalten alle Leitungen des Port B als Eingänge geschaltet. Im Gegensatz zu Ausgängen können Sie nun jedes einzelne Bit auf 0 setzen, indem Sie die entsprechende Leitung mit Masse verbinden. Aber wie läßt sich ein einzelnes Bit vom DPB-Register überprüfen? Des Rätsels Lösung ist die »AND-Verknüpfung«. Um beispielsweise das Bit 7 zu überprüfen geben Sie ein:

```
PRINT PEEK (56577) AND 128
```

Ist das Ergebnis 128, so enthält das Bit 7 des DPB-Registers eine 1. Bei einer 0 als Ergebnis enthält das Bit 7 ebenfalls eine 0. Das Bit 6 können Sie entsprechend mit

```
PRINT PEEK (56577) AND 64
```

überprüfen. Beim Ergebnis 64 enthält das Bit 6 eine 1, bei 0 als Ergebnis ist wieder eine 0 enthalten. Auf die gleiche Weise lassen sich auch alle anderen Bits überprüfen. Was der Computer bei einer AND-Verknüpfung genau macht, können Sie im Sprite-Kurs in diesem Sonderheft nachlesen.

Wenn Sie die beschriebenen Basic-Anweisungen nun in Programme einbinden, können Sie die verschiedensten Steuervorgänge hervorrufen und überprüfen. Alarmanlagen lassen sich ebenso konstruieren wie blinkende Lämpchenreihen oder vieles mehr. Achten Sie aber darauf, daß Sie Ausgängen nur einen sehr begrenzten Strom entnehmen können. Er reicht zwar, um an jeder Leitung eine Leuchtdiode mit Vorwiderstand anzuschließen, aber für weitere Schaltvorgänge sollten Relais oder Verstärker zwischengeschaltet werden.

Natürlich können Sie den User-Port auch für die Datenfernübertragung nutzen und beispielsweise einen Akustikkoppler anschließen. Mehr über dieses Thema können Sie in unserem Datenfernübertragungs-Artikel in diesem Sonderheft erfahren.

So erkennt der C64 Module

Module wie »Simons-Basic« oder das Spiel »Soccer« werden in den Expansion-Port gesteckt. Nach dem Einschalten steht Ihnen dann sofort das entsprechende Programm zur Verfügung. Aber wie erkennt Ihr Computer, daß Sie ein Modul eingesteckt haben? Dafür müssen wir uns die Funktion von einigen Kontakten des Expansion-Ports verdeutlichen.

Das Bild 5 zeigt die Kontakte des Expansion-Ports, wie Sie sie von hinten sehen. Das Modul legt beim Einschalten die Leitung EXROM (Pin 9) gegen Masse (Pin 1,22,A und Z). Dadurch wird ein Speicherbereich von 8 KByte (=8192 Byte) aus dem Speicher des C64 ausgeblendet. Es sind die Speicheradressen 32768 bis 40956. Der Speicherinhalt des eingesteckten Moduls befindet sich jetzt in diesem Bereich. Ihr Computer weiß damit aber noch nicht, daß dort ein Programm beginnt. Diese Information erhält er, wenn nach dem Einschalten die sogenannte »Reset-Routine« durchlaufen wird. Der Computer überprüft nämlich dabei den Inhalt der Speicherzellen 32772 bis 32776. Ist dort der Text »CBM80« enthalten, so weiß der Computer, daß ein Modul eingesteckt ist. Wenn Sie den eingeblendeten Speicherbereich mit den Speicherzellen für »CBM80« vergleichen, dann erkennen Sie, daß der Text im Modul enthalten ist. Ist kein Modul vorhanden, sind die entsprechenden Speicherzellen

nach dem Einschalten leer und der Computer setzt die Reset-Routine fort. Findet der Computer allerdings »CBM80«, so holt er sich aus den ersten beiden eingeblendeten Speicherzellen (32768 und 32769) die dort abgelegte Startadresse des Modulprogramms und springt an die entsprechende Adresse. Das Modulprogramm wird auf diese Weise sofort nach dem Einschalten des Computers gestartet. Natürlich erfolgt auch nach jedem Reset die Abfrage, ob ein Modul vorhanden ist oder nicht. Die einfachste Weise, ein Reset auszulösen, ist übrigens, am User-Port die Kontakte 3 (Reset) und 1 (Masse) miteinander zu verbinden. Mit einem Taster können Sie ein hardwaremäßiges Reset fest vorsehen. Aber beachten Sie die Garantiebedingungen des Herstellers.

In unserem Port-Artikel wollten wir Sie mit einigen grundsätzlichen Verwendungen des Expansion- und User-Ports vertraut machen. Wenn Sie sich noch mehr für die beiden Ports interessieren, dann möchten wir Sie auf einige Artikel in unseren 64'er-Stammheften aufmerksam machen:

Im Artikel »User-Port-Display« (Ausgabe 5/85, Seite 36) erfahren Sie, wie Sie am User-Port LEDs, Relais und Optokoppler anschließen. Kleine Steuerprogramme sind auch dabei.

In »Nicht nur ein Geheimdienst: CIA« (Ausgabe 2/86, Seite 93) finden Sie detaillierte Informationen über den User-Port-Baustein 6526 und dessen Programmierung. Auch wie Sie zwei C64 über den User-Port koppeln, wird beschrieben.

Die 64'er Extra-Seiten enthalten in der Ausgabe 4/86 eine Intensiv-Information über den Expansion-Port – besonders geeignet für alle, die den Expansion-Port einmal hardwaremäßig nutzen wollen.

(kn)



Die seltsamen Wege nach draußen

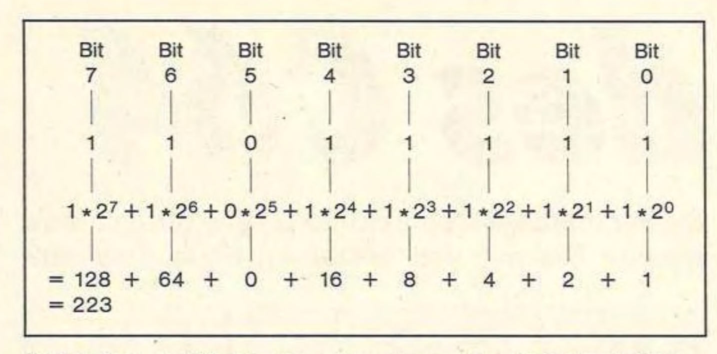


Bild 4. Beispiel für die Umrechnung von Dual-/Dezimalzahlen



Krokodilklemmen dürfen niemals am User-Port angeschlossen werden.

1 2 3 4 5 6 7 8 9 10 11 12

A B C D E F H J K L M N

PIN	BELEGUNG	PIN	BELEGUNG
1	GND	A	GND
2	+5V, max. 100 mA	B	FLAG2
3	$\overline{\text{RESET}}$	C	PB0
4	CNT1	D	PB1
5	SP 1	E	PB2
6	CNT2	F	PB3
7	SP 2	H	PB4
8	$\overline{\text{PC2}}$	J	PB5
9	SER. ATN IN	K	PB6
10	9V AC, max. 100 mA	L	PB7
11	9V AC, max. 100 mA	M	PA2
12	GND	N	GND

Bild 3.
Der User-Port von hinten gesehen und die Kontaktbelegung



Bild 5. Der Expansion-Port von hinten gesehen





Der Massenspeicher mit Intelligenz

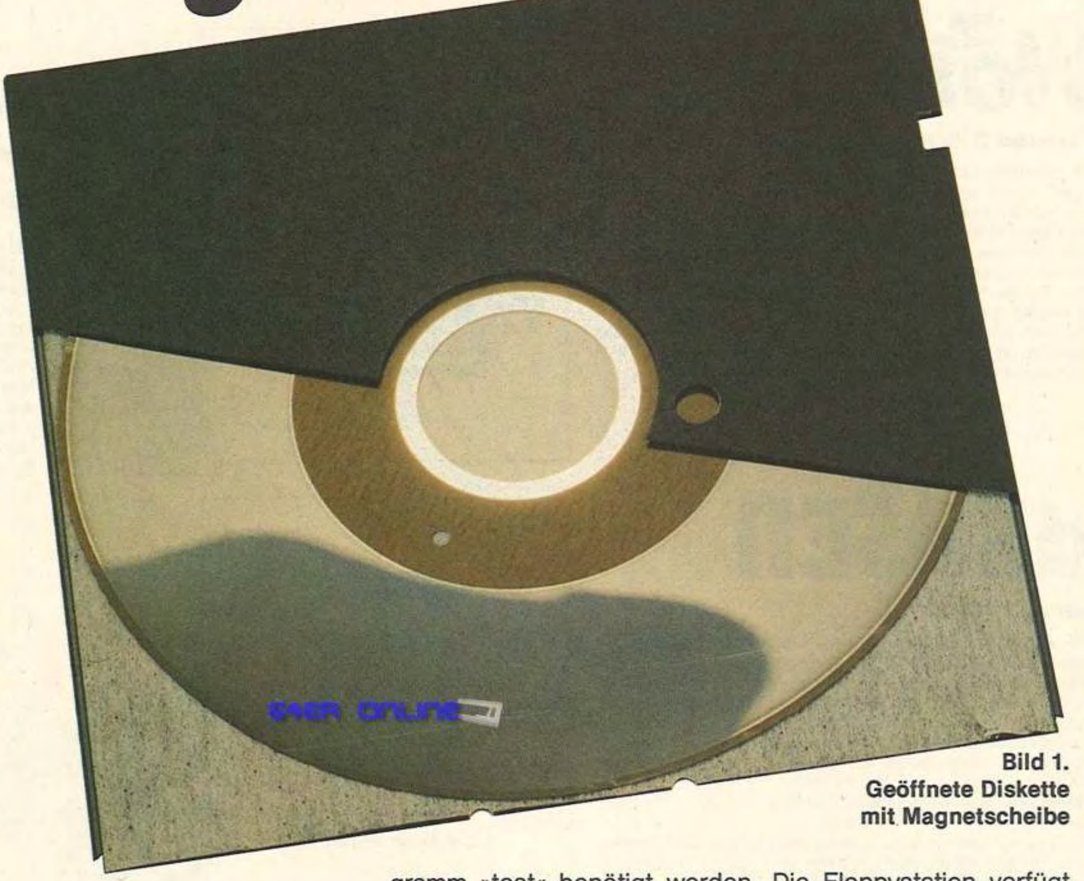


Bild 1. Geöffnete Diskette mit Magnetscheibe

Das Speichermedium, das Ihnen die Arbeit mit Daten und Programmen erleichtert, ist die Diskette. Auf einer runden, flexiblen Magnetscheibe können fast 170 000 Byte gespeichert und verwaltet werden. Das Wie zeigt Ihnen dieser Artikel.

Das Herz einer Diskette ist die Magnetscheibe. Auf einem runden Trägermaterial ist eine magnetische Schicht aus feinen Eisenteilchen aufgetragen (Bild 1). Diese magnetisierbare Schicht kann Magnetzustände speichern, das heißt, die einzelnen Bytes werden in der Magnetschicht eingefroren. Aber ohne Organisation gäbe es ein wildes Chaos, wenn die Daten beliebig auf der Diskette gespeichert würden. Deshalb muß vor dem Gebrauch erst eine Organisation aufgebaut werden. Dieses Organisieren nennt man Formatieren. Mit Hilfe eines Programms wird die Diskette (beim Laufwerk 1541, Bild 2) in 35 Kreisringe (1 bis 35), die Spuren, unterteilt (Bild 3). Auf diese Spuren werden die Informationen in kleineren Bereichen, den Sektoren, untergebracht (Bild 4). Man muß sich das wie einen Karteikasten vorstellen. Eine Kartei ist in einzelne Buchstaben unterteilt, von A bis Z. Unter jedem Buchstaben sind wieder mehrere Karteikarten mit Notizen abgelegt. Das Register A bis Z können wir mit den Spuren vergleichen, die zusätzlichen Karteikarten mit den Sektoren auf jeder Spur (Bild 5).

Zur Verdeutlichung des Prinzips schauen wir uns den Ablauf beim Laden eines Programms an. Der erste Schritt ist die Eingabe des Befehls

```
LOAD "test",8.
```

Der Befehl LOAD veranlaßt den Computer, ein Programm mit dem Namen »test« zu laden. Die Acht hinter dem Komma steht für das Laden von der Floppystation.

Der Computer öffnet den Verbindungskanal zur Floppystation und teilt der Floppy mit, daß die Daten für das Programm »test« benötigt werden. Die Floppystation verfügt über einen eigenständigen Computer, dem DOS = Disk Operating System, der für die Verwaltung aller Daten auf der Diskette verantwortlich ist. Das DOS schaut in einem Register, dem Directory nach, ob Daten unter dem Namen »test« vorliegen. Dieses Directory (Inhaltsverzeichnis), in dem alle Programmnamen gespeichert sind, befindet sich auf jeder Diskette in der Spur 18. Findet das DOS einen Eintrag unter dem Namen »test«, so liest es Daten, die neben dem Namen im Directory gespeichert sind. Diese Daten geben Auskunft darüber, an welcher Stelle (Spur und Sektor) auf der Diskette das Programm beginnt und wieviele Sektoren von dem Programm belegt werden. Das Steuerungssystem fährt den Lesekopf an den Anfang der Information, liest die Daten blockweise aus und übergibt sie dem Computer. Dabei kann jeder Sektor nur eine begrenzte Anzahl an Zeichen speichern (254 Byte). In der Regel sind Programme aber länger als 254 Byte, deshalb müssen die Daten in mehreren Sektoren untergebracht werden. Damit das DOS weiß, in welchem Sektor die Fortsetzung steht, werden jedem Sektor zwei Zeiger vorangestellt, in dem die Adresse des nächsten Sektors gespeichert ist. Sind aus einem Sektor alle Daten ausgelesen, so springt der Lesekopf auf den Sektor mit der Fortsetzung und liest weiter. Das Ende wird dem DOS dadurch angezeigt, daß eine Spur angesprochen wird, die nicht auf der Diskette existiert, die Spur 0. Nachdem die Floppystation alle Daten an den Computer übergeben hat, geht sie wieder in Wartestellung, bis ein neuer Befehl anliegt.

Als Nächstes wollen wir uns den umgekehrten Weg ansehen, das Speichern eines Programms. Zuerst wird wieder

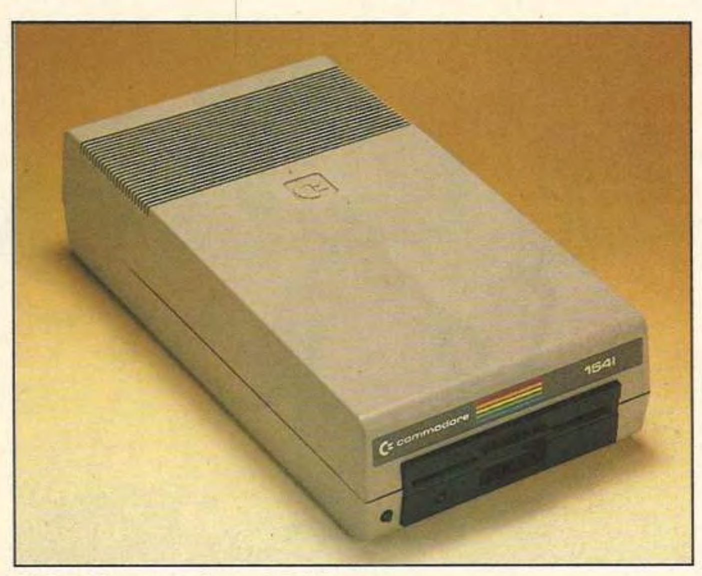


Bild 2. Floppystation mit Disketten

ein Befehl eingegeben, der den Computer veranlaßt, Daten an die Floppystation zu übergeben.

```
SAVE"test",8
```

Nachdem die Floppystation den Befehl erhalten hat, schaut sie nach, wo auf der Diskette noch Platz vorhanden ist. Hat sie ausreichend Platz gefunden, fährt sie den Schreibkopf auf die richtige Position und beginnt die Daten zu speichern. Reicht ein Sektor für die Daten nicht aus, so verkettet sie den Sektor mit anderen freien Sektoren, bis das Programm auf der Diskette untergebracht ist.

Es war bisher des öfteren vom Directory die Sprache. Was ist das Directory? Wir können es mit einem Karteikasten zur Verwaltung von Akten vergleichen (Bild 5). Auf der Karteikarte steht zum Beispiel, daß das Programm »Menütechnik« im X'ten Aktenschrank und unter dem Y-Aktenzeichen zu finden ist. Die Sekretärin braucht dann nur den richtigen Aktenschrank (Spur) öffnen und die Akte mit dem gesuchten Aktenzeichen (Sektor) entnehmen. Genauso verfährt das Directory in unserer Diskettenstation. Neben dem Programmnamen finden wir Einträge über die Spur und den Sektor, bei dem das Programm beginnt. Außerdem befindet sich noch ein Eintrag im Directory, der über die Länge des Programms, das gespeichert ist, Auskunft gibt. Das Directory liegt, wie schon erwähnt, auf der Spur 18, in den Sektoren Eins bis 17. Pro Sektor können wir 8 Programme verwalten. 18 Sektoren mit je acht Programmen ergeben maximal 144 Programme pro Diskette, die abgelegt werden können. Vorausgesetzt ist, daß die einzelnen Programme nicht zu lang sind. Für die 144 Programme stehen Ihnen insgesamt 664 Sektoren zur Verfügung. Woher weiß die Floppystation, wieviel Platz noch übrig ist? Dazu befindet sich im Sektor Null der Spur 18 eine Einrichtung, die die Belegung von Spuren und Sektoren beinhaltet, die BAM. BAM steht für Block Availability Map = Sektor-Belegungs-Plan. In dieser BAM ist für jeden Sektor auf der Diskette ein Bit reserviert, das Auskunft darüber gibt, ob dieser Sektor mit Daten belegt ist oder nicht. Ist ein Sektor belegt, so steht hier eine Null. Für jeden freien Block steht eine Eins.

Die eingebaute Müllabfuhr

Was macht die Floppystation mit Programmen und Daten die nicht mehr benötigt werden und noch Teile der Diskette belegen? Dazu gibt es in der Floppystation eine Art Müllabfuhr, den SCRATCH-Befehl. Geben wir den Befehl

```
OPEN 15,8,15,"S:NAME":CLOSE 15
OPEN 15,8,15,"S:F*":CLOSE 15
OPEN 15,8,15,"S:*":CLOSE 15
```

Syntax:

OPEN	= > Mit OPEN eröffnen wir einen Verbindungskanal zu einem Gerät (Datasette, Floppystation, Drucker) zur Übertragung von Daten. Auch als Öffnen eines File bezeichnet.
15	= > Mit der ersten Ziffer teilen wir dem Gerät eine logische Nummer (Filenummer) zwischen 0 und 127 zu.
8	= > Mit der zweiten Ziffer legen wir das Gerät fest (0 Tastatur, 1 Datasette, 3 Bildschirm, 4 bis 7 Drucker). In unserem Fall steht die 8 für die Floppystation.
15	= > Mit der dritten Ziffer legen wir den Übertragungskanal fest. Dabei sind: 0 für das Lesen eines Programms von der Floppy, 1 für das Speichern eines Programms auf der Floppy, 2 bis 14 ist für diverse Zwecke, 15 ist zur Befehlsübertragung an die Floppy reserviert.
S:	= > Das S steht für SCRATCH = löschen.

Mit dem ersten Befehl löscht man das Programm mit der Bezeichnung »Namen« aus dem Directory. Damit ist das Programm auf der Diskette nicht mehr präsent (daß dieses nicht ganz stimmt, sehen wir später). Mit dem zweiten Befehl löscht man alle Programme, die mit F beginnen. Dabei steht das Sternchen für einen Joker. (Alle Zeichen, die ab dem Joker im Programmnamen stehen, werden vernachlässigt.) Mit dem dritten Befehl löscht man alle Programme, die auf der Diskette sind. Aber ist damit das Platzproblem gelöst? Ja und nein. Das Programm ist zwar aus dem Directory verbannt, aber in der BAM sind die Sektoren noch als belegt gekennzeichnet. Dieses läßt sich mit dem folgenden Befehl schnell nachholen: `OPEN 15,8,15,"V:".`

Syntax:

OPEN	= > Öffnen eines Files
15	= > logische Filenummer
8	= > Geräteadresse
15	= > Befehlskanal
V:	= > Validate = Aufräumen

Mit Validate wird ein Programm aufgerufen, das überprüft, welche Sektoren noch mit einem Programm belegt sind. Alle als frei erkannten Sektoren werden mit einer Eins in der BAM gekennzeichnet. Es gibt noch eine weitere Möglichkeit, die Diskette aufzuräumen. Aber Vorsicht, dieser Befehl kann nicht mehr rückgängig gemacht werden. Dagegen ist es bei

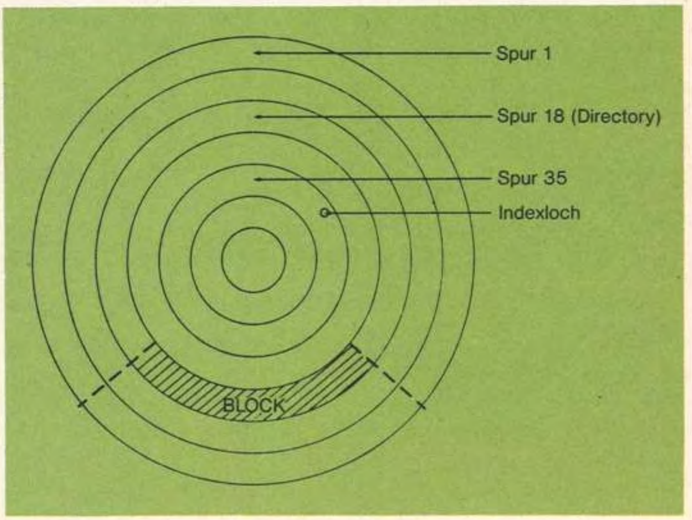


Bild 3. Aufteilung der Diskette in die 35 Spuren und Sektoren

den bisher genannten Aufräummethoden mit einem speziellen Programm noch möglich gewesen.

Die radikale Methode ist das Formatieren. Dabei gibt es zwei Formatierungsbefehle, das kurze und das lange Formatieren.

```
OPEN 15,8,15,"N:Diskettenname":CLOSE 15
OPEN 15,8,15,"N:Diskettenname,ID":CLOSE 15
```

Syntax:

OPEN	=> Öffnen eines Files
15	=> logische Filenummer
8	=> Geräteadresse
15	=> Befehlskanal
N:	=> Befehl für NEW = Neu
,ID	=> Identifikationsmarke, bestehend aus zwei Buchstaben, aus der die Floppystation erkennt, ob die Diskette zwischenzeitlich gewechselt wurde.



Den langen Formatierungsbefehl kennen Sie bereits, wenn Sie schon über eine Floppystation verfügen, jedesmal wenn Sie eine neue, kauffrische Diskette zum ersten Gebrauch vorbereiten. Hierbei werden die Spuren und Sektoren neu angelegt und alle Datenblöcke überschrieben. Nach dieser Befehlsausführung sind keine alten Daten mehr auf der Diskette zu finden. Läßt man bei dem Formatierungsbefehl die ID-Kennung weg, so hat man einen kurzen Formatierungsbefehl. Bei diesem Befehl wird nur das Directory und die BAM neu angelegt. Den Unterschied erkennt man schon bei der Länge der Befehlsausführung. Das lange Formatieren benötigt so um eine Minute, bis alle Sektoren und Spuren neu angelegt sind. Der kurze Formatierungsbefehl ist schon nach einigen Sekunden abgearbeitet.



Gelöscht und doch nicht gelöscht

Beim kurzen Formatieren und beim Scratch-Befehl sind die alten Daten noch in den Sektoren erhalten geblieben. Nur im Directory und in der BAM sind diese Sektoren als frei gekennzeichnet worden. Also besteht die Möglichkeit, diese Daten zu restaurieren. Dafür gibt es spezielle Programme, wie den Diskmonitor. Aber keine Angst, wir gehen nicht weiter darauf ein. Es ist nur wichtig zu wissen, daß versehentlich gelöschte Daten auf der Diskette gerettet werden können. Deshalb verweise ich auf ein Buch von Karsten Schramm, »Die Floppy 1541«, Markt & Technik Verlag, ISBN 3-89090-098-4, Preis 49 Mark, in dem Sie mehr über die Floppystation lesen können.

Error, Error und noch ein Error

Eine Meldung, die schon des öfteren den Anfänger zum Nägelkauen, Haareraufen oder zum Aufschreien veranlaßt hat, ist eine Fehlermeldung des Computers. Beim Gebrauch der Floppy kommen noch weitere spezielle Fehlermeldungen der Floppystation hinzu. Ich werde Ihnen einige Ursachen und deren Behebung nennen. Dazu einige Beispiele, die Sie ausprobieren sollten.

Wir beginnen mit den Meldungen des Computers. Löschen Sie Ihren Programmspeicher mit NEW und geben folgendes Programm ein:

```
20 PRINT#8,"TEST"
```

Nach der Eingabe von RUN meldet sich der Computer mit der Fehlermeldung »File not open error in 20« Bevor eine Printanweisung an ein Gerät gegeben werden kann, muß die

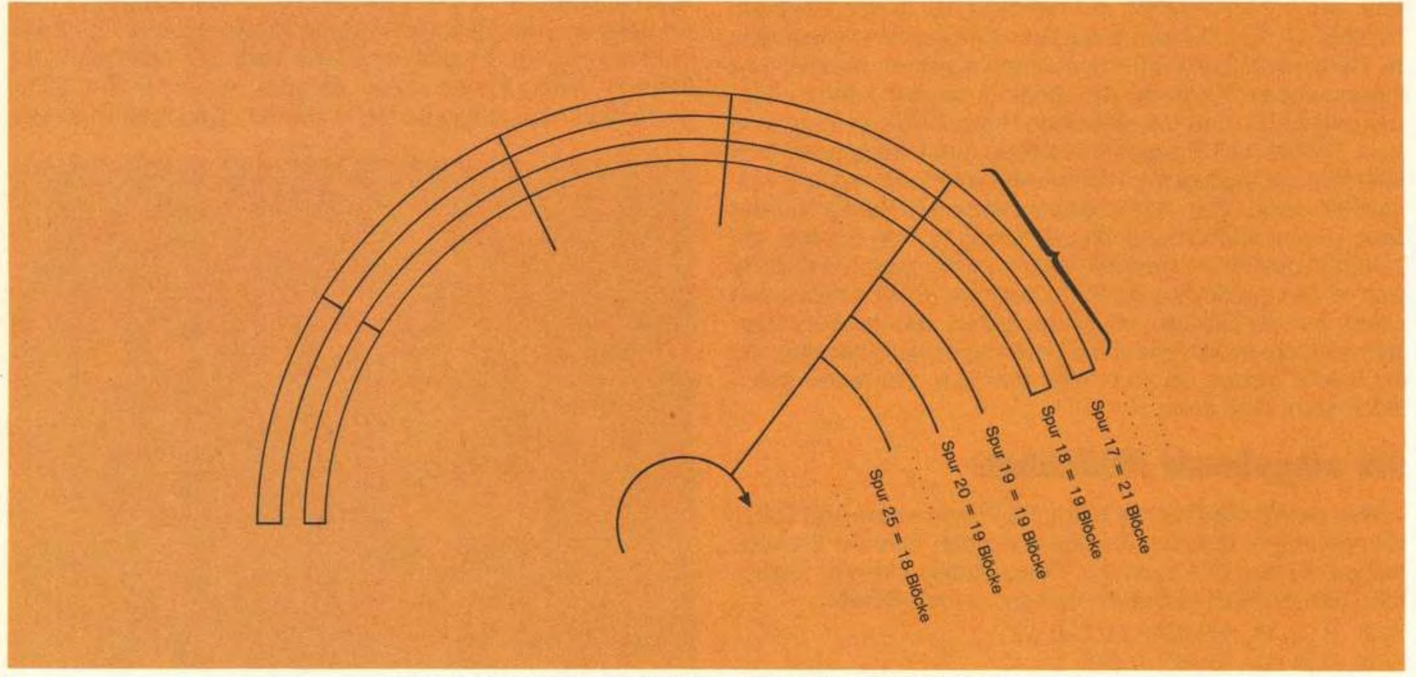


Bild 4. Verteilung der Sektoren (Blöcke) auf der Diskette

Verbindung dorthin erst eröffnet werden. Dieses erreichen wir mit der folgenden Ergänzung unseres Programms.

```
10 OPEN 8,8,2,"TEST,W"
```

Bevor Sie dieses kleine Programm starten, schalten Sie bitte die Floppystation aus. Nach RUN erhalten Sie folgende Meldung: »Device not present error in 10«. Und das ist auch richtig so, denn das angesprochene Gerät ist ja nicht aktiv und damit nicht vorhanden. Die gleiche Meldung erhalten Sie auch, wenn Sie die Gerätenummer verwechseln, so zum Beispiel: die Floppystation mit OPEN 8,4,2 ansprechen. Die Geräteadresse 4 (zweite Ziffer) ist für den Drucker reserviert. Wir schalten die Floppy wieder ein und starten das Programm neu. Jetzt führt der Computer unsere Anweisung aus. Aber nun brennt die LED-Anzeige (Luminizenzdiode) an der Floppy permanent. Ein Zeichen, daß die Floppy noch aktiv ist. Dieses beheben wir mit dem Befehl

```
90 CLOSE 8
```

Mit dem Befehl »CLOSE« schließen wir den eröffneten Gerätekanal wieder. Starten Sie das Programm neu und sehen Sie, was passiert. Zu Ihrem Erstaunen wird nun die LED wild blinken. Warum? Das Blinken bedeutet, daß ein Fehler in der Floppy aufgetreten ist. Um diesen Fehler zu finden, müssen wir die Floppy fragen, welcher Fehler vorliegt. Dazu hilft uns folgendes kleines Programm:

```
100 GOSUB 10000 :REM UNTERPROGRAMMAUFRUF ZUR
ABFRAGE DES FEHLERCODES
9999 END
10000 OPEN 15,8,15 :OEFFNEN DES BEFEHLSKANALS
10010 INPUT#15,EN,EM$,ET,ES
10020 PRINT"FEHLERNUMMER:";TAB(20)EN
10030 PRINT"FEHLERMELDUNG:";TAB(21)EM$
10040 PRINT"FEHLER IN SPUR:";TAB(20)ET
10050 PRINT"FEHLER IN SEKTOR:";TAB(20)ES
10999 CLOSE 15 :RETURN
```

Die Zeilen haben folgende Bedeutung:
100 verzweigt in die Abfrageroutine.
10000 eröffnet den Befehlskanal zur Floppy
10010 fragt die Parameter ab und speichert sie in Variablen.
10020 bis 10050 gibt die Meldungen mit einem Kommentar auf dem Bildschirm aus.

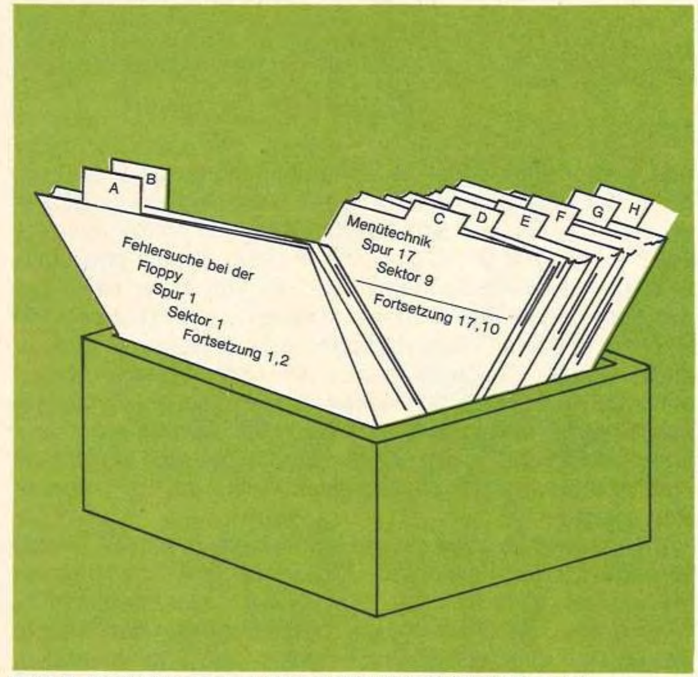


Bild 5. Verwaltung von Akten mit einem Karteikasten

10999 schließt den Befehlskanal wieder und springt zurück ins Hauptmenü.

Diese kleine Routine sollte in jedem Programm, das eine Floppystation aufruft, eingebaut sein.

Haben Sie die kleine Ergänzung eingegeben, so starten Sie wieder mit RUN. Nun sollte, wenn Sie das Programm fehlerfrei eingetippt haben, folgendes auf dem Bildschirm erscheinen:

```
Fehlermeldung     63
Fehlermeldung     file exist
Fehler in Spur    0
Fehler in Sektor  0
```

Kommen wir zu unserem Fehler »file exist« zurück. Im dritten Test haben wir eine Datei mit dem Namen »test« eröffnet. Im letzten Versuch versuchten wir eine weitere Datei mit dem gleichen Namen zu eröffnen. Zwei Einträge mit dem gleichen Namen sind aber nicht zulässig. Deshalb gibt uns die Floppystation eine Fehlermeldung aus und erwartet, daß wir den Namen ändern. Laut Handbuch für die Floppystation besteht die Möglichkeit, mit den Klammeraffen @ im Namen eine Datei zu überschreiben. Leider ist in der Software ein Fehler enthalten, der zum Verlust Ihrer Daten führen könnte. Im folgenden Programm kann dieses nicht mehr passieren.

```
5 INPUT"{SHIFT/HOME}DATEI ZUM SCHREIBEN OEFFNEN
TEST{7*CURSOR LEFT}";N$
10 OPEN 8,8,2,N$+",S,W"
15 GOSUB 10000
20 PRINT#8,"TEST"
90 CLOSE 8
100 GOSUB 10000 :REM UNTERROUTINE ZUM ABFRAGEN DER
FEHLERCODES
9999 END
10000 OPEN 15,8,15 :OEFFNEN DES BEFEHLKANALS
10010 INPUT#15,EN,EM$,ET,ES
10020 PRINT"{4*CURSOR DOWN}FEHLERNUMMER:
";TAB(20)EN
10030 PRINT"FEHLERMELDUNG:";TAB(21)EM$
10040 PRINT"FEHLER IN SPUR:";TAB(20)ET
10050 PRINT"FEHLER IN SEKTOR:",TAB(20)ES
10090 CLOSE 15
10100 IF EN = 63 THEN CLOSE15:GOTO 11000
10999 RETURN
11000 INPUT"ARE YOU SURE? YES{5*CURSOR
LEFT}";X$:IF X$ = "YES" THEN 11020
11005 INPUT"NEUER PROGRAMMNAME:";N$:
CLOSE 8:OPEN8,8,2,N$+",S,W"
11010 CLOSE 15: GOTO10999
11020 OPEN15,8,15,"S:"+N$:GOTO11010
```

Änderungen:
5 ist eine Eingabe, die den Dateinamen in eine Stringvariable wandelt.
10 Der Name ist durch die Variable ersetzt. Um die Variable universell zu halten, wird der Schreibbefehl erst in dieser Zeile mit den Namen (+",S,W") verbunden.
10100 fragt ab, ob der Fehler 63 (file exist) vorliegt. Wenn ja, dann wird in die Zeile 11000 verzweigt.
11000 folgt eine Sicherheitsabfrage ob die Daten gelöscht werden sollen. Wenn ja, wird in die Zeile 11020 verzweigt.
11005 wird ein neuer Name eingegeben und die Datei neu eröffnet.
11020 werden die alten Daten gelöscht.

Weitere Hinweise und Beispiele zur Dateiverwaltung finden Sie in dem Artikel »Dateiverwaltung für Einsteiger«.

(do)

Wie kommt die Bewegung vom Joystick in den Computer?

Wie genial und einfach das Prinzip des Joysticks ist, läßt sich von außen kaum erahnen. Von der Mechanik bis zum Spiel ist es gar nicht so weit, wie Sie glauben. Deshalb wollen wir Ihnen hier erklären, wie ein Joystick funktioniert.

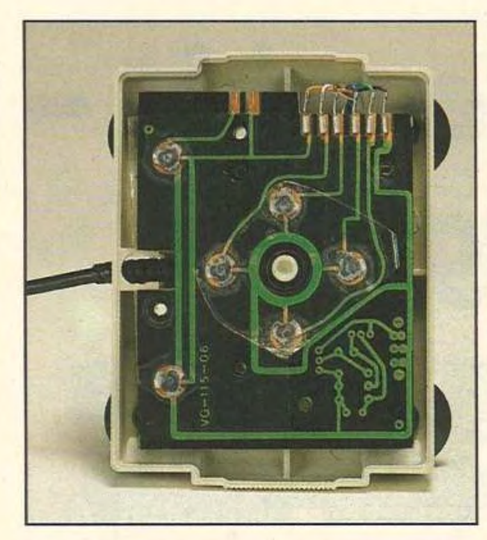


Bild 5. Nach diesem Prinzip sind die meisten Joysticks verdrahtet.

Heiße Actionszenen und punktgenaue Zeichenroutinen lassen sich durch den Joystick hervorragend realisieren. Die Umsetzung der Bewegung der Hand in die Bewegung eines Sprites oder eines Cursors wird durch den Joystick erheblich vereinfacht. Dahinter verbirgt sich jedoch längst nicht so viel Aufwand, wie sich vermuten läßt. Aufwendige elektronische Schaltungen oder komplizierte Mechaniken gibt es dort nicht, selbst der Aufwand der Software hält sich in Grenzen. Um Sie mit dem Eingabegerät Joystick vertraut zu machen, haben wir für Sie den Weg vom Joystick in den Computer nachvollzogen.

Rein äußerlich betrachtet besteht ein Joystick aus dem Gehäuse, einem beweglichen Griff und ein bis drei Feuerknöpfen. Würden Sie Ihren Joystick öffnen, dann würden Ihnen zuerst vier Schalter auffallen (Bild 1 bis 4). Sie sind so angeordnet, daß entweder der durch den Mittelpunkt ragende Hebel oder ein mit Noppen versehener Ring (Bild 1 und 4) beim Bewegen jeweils einen oder, in der Diagonale, zwei Kontakte schließt. Dies kann der Computer erkennen und entsprechend seines Programmes darauf reagieren. Eben diese Kontakte sind übrigens entscheidend für Stabilität, Lebensdauer und Schaltgefühl. Als anfällig haben sich die Folienkontakte in Bild 1 erwiesen. Es sind einfach zwei übereinander angeordnete Metallplättchen, von denen eines von den obengenannten Noppen durchgedrückt wird und so den Kontakt herstellt. Dieses Prinzip entspricht dem der Folientastatur einiger Taschenrechner. Die Konstruktion verliert schnell ihre anfängliche Festigkeit und leiert bei Belastung aus. Besser (und teurer) sind da schon Metallzungenkontakte (Bild 2) und Mikroschalter (Bild 3). Sie sind wesentlich stabiler und widerstehen intensivem Gebrauch und stärkster Belastung.

Die Schalter im Joystick sind nach einem einfachen Prinzip verdrahtet (Bild 5). Selbst das Dauerfeuer läßt sich mit einem kleinen, 8poligen IC (zum Beispiel der Timer-Baustein NE 555) realisieren. Die aus dem Joystick kommenden Signale lassen sich über den Joystick-Port an den Speicherstellen 56320 (Port 2) und 56321 (Port 1) auslesen.



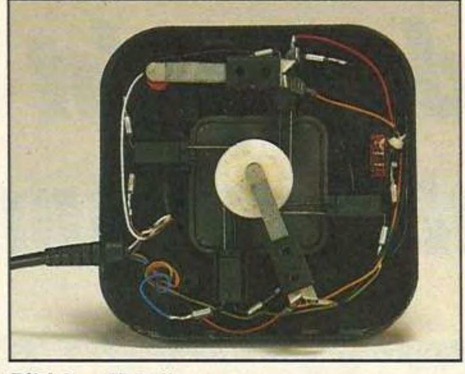


Bild 2. ...Metallzungen,...

Bild 3. ...Mikroschaltern,...

Bild 1. So funktioniert ein Joystick. Entweder mit Folienkontakten,...

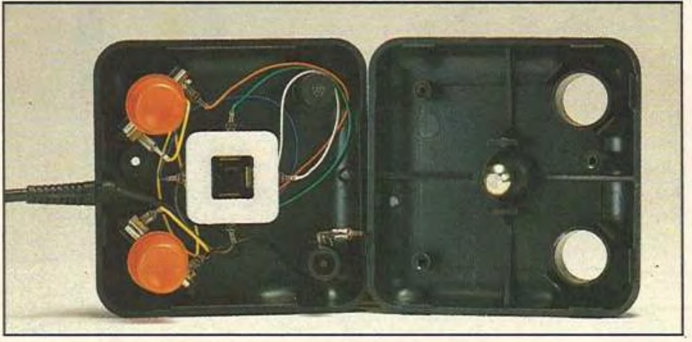
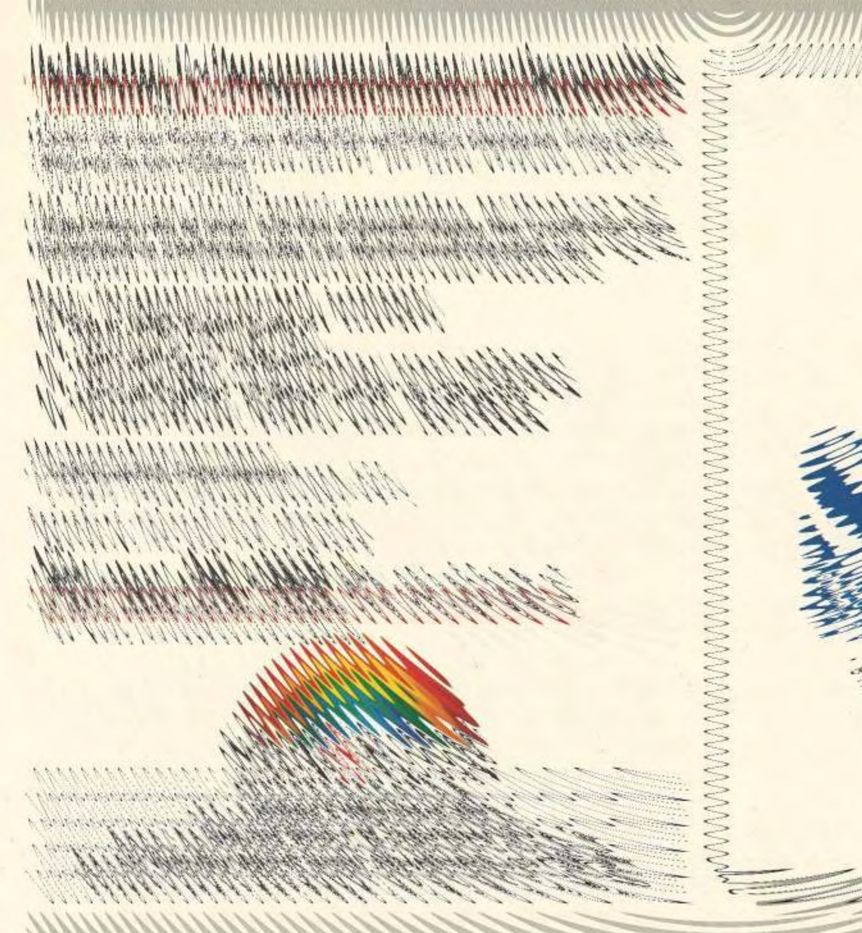
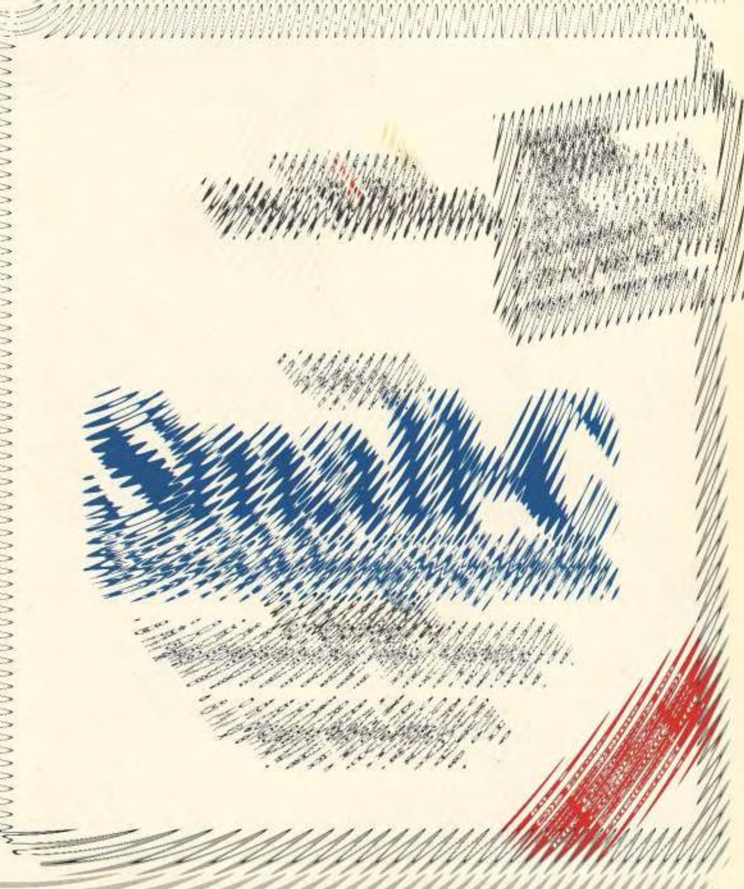


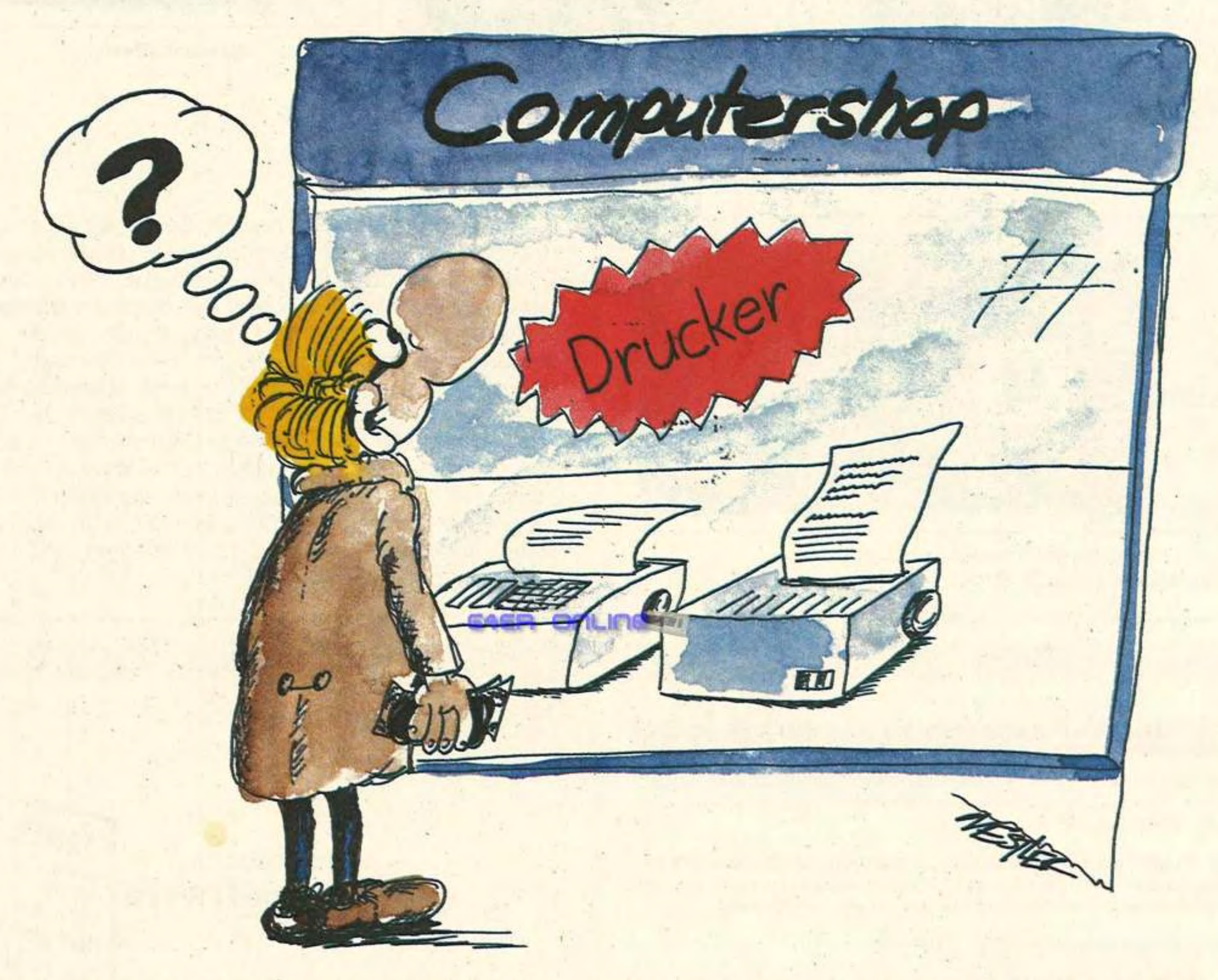
Bild 4. ...oder Kugel-Kontakt-Schaltern. Hauptsache, es fließt Strom.

Dort sind die entsprechenden Leitungen in einem Byte zusammengefaßt (siehe Sprite-Kurs in dieser Ausgabe und »Der Trick mit dem Joystick«, Ausgabe 11/85, Seite 24). Zwischen Port und Basic liegt nur noch ein Baustein, CIA (Complex Interface Adapter) genannt, der fast die gesamte Steuerung externer Geräte im C64 verwaltet. Dieser Interface-Baustein übermittelt nun beim Ausleseversuch der Adressen 56320 oder 56321 die am entsprechenden Joystick-Port anliegenden Signale an den Prozessor. Damit steht dem Computer die gewünschte Information zur Verfügung. Das war es dann auch schon, denn ab hier ist alles eine Frage der Software, also Joystick links bewirkt die Bewegung einer Spielfigur nach links, Feuerknopf drücken und die Figur nimmt einen Gegenstand auf, etc. (og)





Ohne Drucker geht es nicht – eine Entscheidungshilfe



Ein Computer ohne Drucker ist wie ein Füller ohne Tinte – man kann damit schreiben, aber es bleibt nichts auf dem Papier zurück. Der Drucker ist die sinnvollste Erweiterung Ihres Computersystems, doch eine Entscheidung zwischen den vielen Leistungsdaten, Preisen und Normen zu treffen ist schwierig. Deshalb helfen wir Ihnen.

Klar, Sie brauchen einen Drucker – nur welcher ist der richtige? Selbstverständlich möchten Sie auch das Beste für Ihr Geld bekommen und das Risiko des Fehlkaufs möglichst gering halten. Aber sicherlich kennen Sie auch schon den »Verwirrungs-Effekt«, wenn Sie sich in verschiedenen Kaufhäusern und Fachgeschäften informiert oder den Anzeigenteil der 64'er gelesen haben. Es ist verständlich und legitim, daß jeder Hersteller sein Produkt für das einzig »seligmachende« hält, denn sonst würde er es wohl nicht verkaufen. Mit den Augen des Konsumenten sieht die Sache allerdings ganz anders aus, denn fast jeder will mit seinem Drucker etwas anderes machen. Dementsprechend sollte dann auch die Wahl des entsprechenden Druckers ausfallen. Wie Sie dabei vorgehen können, wollen wir Ihnen hier beschreiben und anschließend einige Drucker vorstellen, die unseren Tests nach empfehlenswert sind.

Fassen wir doch einmal zusammen, welche Kriterien für den Druckerkauf von grundlegender Bedeutung sind, alle anderen Entscheidungen hängen davon ab und werden erst viel später wichtig. Im wesentlichen sind es nur vier Kriterien, die es zunächst zu beachten heißt.

1. Das Schriftbild

An erster Stelle steht natürlich das Schriftbild. Da das Schriftbild aber im wesentlichen vom Druckprinzip abhängig ist, wollen wir es auch in diesem Zusammenhang erklären. Generell sollte ein guter Drucker heute mindestens über richtige Unterlängen (zum Beispiel bei den Buchstaben »g« und »p«) verfügen. Die Buchstaben sollten möglichst wohlgeformt sein und, ganz besonders wichtig, deutsche Umlaute dürfen nicht fehlen. Das beste Schriftbild haben heute immer noch die Typenraddrucker, die im Prinzip nichts anderes sind als eine Schreibmaschine ohne Tastatur. Dafür besitzen Typenraddrucker einen Computeranschluß, den man auch



als Schnittstelle bezeichnet. Ein gleich gutes Schriftbild haben übrigens auch Computerschreibmaschinen, die sowohl eine Tastatur als auch eine Computer-Schnittstelle besitzen. Leider sind sowohl Computerschreibmaschinen als auch Typenraddrucker sehr langsam, sehr laut und mitunter auch recht teuer, außerdem sind sie nicht in der Lage, Grafiken (die wir in Punkt 2. beschreiben werden) auszudrucken.

Das zweitbeste Schriftbild haben mittlerweile die Matrixdrucker. Matrixdrucker heißen so, weil bei ihnen die Schrift aus einzelnen Punkten zusammengesetzt wird. Zum Drucken der einzelnen Punkte werden drei verschiedene Verfahren angewendet: Das erste Verfahren heißt Tintenstrahl-Verfahren, denn die einzelnen Punkte werden in Form von mikroskopisch kleinen Tropfen auf das Papier geschleudert. Tintenstrahldrucker sind sehr leise, aber leider auch sehr teuer. Bei älteren Modellen kommt es auch immer noch vor, daß die Tinte eintrocknet und eine teure Wartung notwendig wird. Dies sind auch die Gründe, warum es noch sehr wenige Drucker dieser Art gibt. Neuere Entwicklungen deuten aber an, daß die Zukunft diesem Verfahren in zunehmenden Maße gehören wird.

Das zweite Verfahren heißt Thermo- oder auch Thermo-Transfer-Verfahren. Bei ihm werden die einzelnen Punkte, aus denen die Buchstaben zusammengesetzt werden, durch die Erhitzung feinster Punkte entweder direkt auf speziellem (teurem) Papier erzeugt oder von einem wachsartigen Farbband abgeschmolzen. Thermodrucker sind sehr leise, aber technisch bedingt auch relativ langsam, denn der Druckkopf muß zwischen den Zeichen immer etwas abkühlen. Trotzdem erzeugen Thermodrucker der neueren Generation ein ganz ausgezeichnetes Schriftbild. Leider sind die Unterhaltskosten für solche Drucker überdurchschnittlich hoch, denn entweder druckt man ohne ein Farbband direkt auf dem teuren Thermopapier oder man druckt mit dem teuren Farbband auf billigem Normalpapier. Am weitesten verbreitet (und damit auch am empfehlenswertesten) sind die Nadel-Matrixdrucker. So seltsam es klingen mag – diese Drucker arbeiten mit kleinen, stumpfen Nadeln, die durch Elektromagneten nach vorne bewegt werden und dabei das Farbband gegen das Papier drücken. Wie Sie sich vorstellen können, machen diese Nadeln einen gehörigen Krach und es gibt keinen Drucker der nach diesem Prinzip arbeitet, den man überhören könnte. Die neueste Generation dieser Drucker verfügt im Gegensatz zu den älteren Modellen auch über eine besonders schöne Schrift. Diese Schrift nennt man »Near Letter Quality«-Schrift und man will damit ausdrücken, daß man damit auch Briefe schreiben kann, ohne sich dabei für die lausige Schrift entschuldigen zu müssen. Diese Schrift, die man eigentlich treffender als Schönschrift bezeichnen sollte, beruht auf einem raffinierten Trick. Da jeder Drucker ja nur eine begrenzte Anzahl von Drucknadeln besitzt (meistens 9), entstehen zwischen den Punkten auf dem Papier immer kleine Lücken. Wenn man aber zweimal über jeden Buchstaben hinwegdruckt und das Papier dabei genausoweit vorschiebt, daß quasi »Nadel auf Lücke« steht, so werden die Zwischenräume ebenfalls bedruckt und es entsteht ein durchgehendes Schriftbild. Neben den Schriftqualitäten zeichnen sich Matrixdrucker vor allem durch ihre Geschwindigkeit (bis 400 Zeichen/Sekunde) und ihre Flexibilität aus. Das spürt man vor allem dann, wenn man mit seinem Drucker mal etwas anderes als Text drucken möchte. Namentlich ist damit natürlich die Grafik gemeint. Fast alle heutigen Computer wie zum Beispiel der C 64, C 128, aber auch die »kleinen« Commodore-Computer verfügen über eine ganz exzellente Grafik. Es wäre doch schade, wenn diese »Kunstwerke« nur auf dem Bildschirm zu sehen wären. Nun, fast alle Matrixdrucker sind in der Lage, beliebige Grafiken auszudrucken. Dabei besitzen sie sogar die Fähigkeit, zwischen verschiedenen »Punktdichten« zu wählen, das heißt einen unterschiedlichen Abstand zwischen den einzelnen Druckpunkten nebeneinander zu realisieren. Das kann so weit gehen, daß sich die Punkte überlappen und das bis zu 90 Prozent. Natürlich dauert so ein Ausdruck dann recht lange und auch das Farbband muß entsprechend strapaziert werden, um das Beste herzugeben. Das Resultat, nämlich der fertige Ausdruck, entschädigt dann für alle Mühen und Sie werden keine Mark bereuen, die Sie ausgegeben haben.

In letzter Zeit setzen sich immer mehr die Farbdrucker durch. Sie sind in der Lage, nicht nur jede Farbe des Computers korrekt wiederzugeben, sie produzieren sogar richtige Gemälde, wenn man sie mit dem richtigen Programm ansteuert. Farbdrucker arbeiten nach allen drei oben beschriebenen Funktionsprinzipien (Tintenstrahl-, Thermo- oder Matrixdrucker). Ganz besonders flexibel sind dabei die Matrixdrucker mit nachrüstbarer Farboption. Sie verbinden die Vorteile eines vollwertigen Textdruckers mit denen der farbigen Textdarstellung, denn sie lassen sich wahlweise mit einem schwarzen Farbband (kostengünstig) oder mit einem »richtig farbigen« Farbband ausrüsten.

2. Die Grafikfähigkeit

Dieser Begriff geistert durch die Werbeprospekte fast aller Druckerhersteller – dennoch ist Grafik nicht gleich Grafik. Unter diesem Begriff versteht man die Funktion des Druckers, die vom Computer ankommenden Daten nicht mehr als Schriftzeichen zu interpretieren, sondern als Grafikmuster. Es hat auch wenig Einfluß auf die generelle Grafikfähigkeit, ob ein Drucker nur 300 oder 2000 Mark kostet. Wie man so eine Grafik programmiert, haben wir in dieser Ausgabe in einem speziellen Artikel beschrieben. Woran erkennt man aber, ob ein Drucker grafikfähig ist oder nicht? Zunächst einmal am Druckprinzip, denn weder eine Schreibmaschine noch ein Typenraddrucker können diese Fähigkeit besitzen. Bei Tintenstrahl-, Thermo- oder Matrixdruckern ist eine Grafikfähigkeit möglich, aber nicht obligatorisch. Bestes Erkennungszeichen sind die technischen Angaben zu dem betreffenden Drucker, wenn man dort Angaben über Begriffe wie Punktdichte/Zeile (manchmal auch pro Inch) findet, sieht die Sache dann schon ganz gut aus. Bleibt nur zu klären, wie gut die Grafik ist. Dabei gilt, daß die Grafikfähigkeit um so besser ist, je höher die Punktdichte pro Zeile ist. Der Standard liegt zur Zeit bei 1920 Punkten pro Zeile (vierfache Dichte). Superdrucker kommen inzwischen auch schon auf 2880 Punkte pro Zeile. Was unter dem Standard liegt, ist eigentlich nicht mehr zeitgemäß, kann aber im Rahmen einer Preis-Leistungskalkulation durchaus noch Sinn haben.

3. Die Schnittstelle

So einfach es auch klingen mag, es ist nicht ganz einfach, einen beliebigen Drucker an einen Commodore-Computer anzuschließen. Am schnellsten geht es natürlich, einen direkt von Commodore gelieferten Drucker (MPS 801, MPS 802, MPS 803) anzustecken. In diesem Fall genügt es, den Drucker mit einem einfachen Kabel zu verbinden. Leider sind die Leistungen der Commodore-Drucker durchwegs sehr gering oder unausgewogen. So beherrscht der MPS 802 den Textdruck recht ordentlich, versagt aber bei der Grafik. Die MPS 801- und MPS 803-Drucker beherrschen zwar die Grafik, besitzen dafür aber ein unbefriedigendes Schriftbild. Preislich heimsen die Originaldrucker einen leichten Vorteil ein, denn sie können wie gesagt direkt angeschlossen werden, ein Interface zum Anpassen anderer Schnittstellennormen entfällt somit. Fast alle anderen Drucker benötigen dagegen ein Interface, das sich am besten als Dolmetscher beschreiben läßt. So ein Interface setzt nämlich die Daten des Computers so um, daß sie der Drucker (meistens von

einem anderen Hersteller) verstehen kann. Solche, meistens sehr leistungsfähigen (aber nicht unbedingt teuren) Drucker werden von ihrem Hersteller in der Regel mit einer Schnittstelle nach der Centronics-Norm ausgestattet. Im einfachsten Fall wird das notwendige Interface einfach in den Drucker eingebaut und man kann ihn anschließen wie einen Commodore-Drucker. Manchmal geht das aber nicht, dann wird es notwendig, ein externes Interface zu verwenden. In Form eines kleinen Kästchens (Zigarettenschachtel-Format), das in das Verbindungskabel integriert ist, erfüllt so ein Interface meistens seine Aufgabe so, daß Sie gar nicht merken, daß Ihr Drucker nicht von Commodore ist. Der Vorteil dieser Lösung liegt darin, daß Sie Ihren Drucker auch an allen weiteren Computern (die Sie sich möglicherweise noch kaufen wollen) verwenden können, denn Commodore ist einer der letzten Computer-Hersteller, der sich nicht an die Centronics-Norm hält. Andersherum, nämlich der Anschluß eines Commodore-Druckers an zukünftige Computergenerationen, ist dagegen beinahe unmöglich, zumindest aber sehr teuer. Aber auch bei der Wahl des Interfaces gibt es Unterschiede. Hier dürfen Sie jedoch vielen zufriedenen Anwendern trauen, die eines der nachstehend genannten Interfaces erworben haben. Seit langem bekannt ist das Wiesemann Typ 9200-Interface, das es beinahe für jeden Drucker gibt (ab 248 Mark). Es läßt sich leicht bedienen und arbeitet sehr zuverlässig. Zudem gibt es dieses Interface mittlerweile recht günstig, das heißt es wird sogar mit vielen Druckern direkt vom Händler empfohlen und verkauft. Gleichsam sehr leistungsfähig ist das Görlitz VC-Interface (239 Mark), das es für Epson-Drucker sogar in einer Einbauversion gibt. Das dritte bekannte Interface ist das PCB IF64/128 von HDS (248 Mark). Es zeichnet sich durch seine Flexibilität und die einfache Bedienung aus.

4. Der Befehlssatz

Jeder Drucker verfügt, wie ein Computer, über einen eigenen Befehlssatz. Man kann ihn somit auch richtig programmieren. Da ein Drucker natürlich keine Tastatur hat, macht man das über den Computer. Aber so wie es verschiedene Computer mit verschiedenen Leistungen gibt, besitzen auch die Drucker unterschiedliche Leistungsmerkmale. Dabei ist es nicht nur wichtig, daß möglichst viele sinnvolle und leistungsfähige Befehle zur Verfügung stehen, sondern auch die Art, wie die Befehle aufgerufen werden, ist wichtig. Stellen Sie sich vor, Sie entwickeln ein Programm, das viel mit dem Drucker arbeitet. Sie werden das Programm sicherlich an Ihren Drucker anpassen. Was macht aber ein professioneller Software-Entwickler, der sein Programm ja für möglichst viele Drucker schreiben möchte – er sucht einen Standard-Befehlssatz, an den sich die meisten Drucker halten. Solch einen Befehlssatz gibt es, er hat den seltsamen Namen ESC/P (Epson Standard Code for Printers) und wurde von der Firma Epson definiert, da deren Drucker-Befehlssatz ohnehin schon als Industriestandard galt. Achten Sie deshalb darauf, daß das von Ihnen favorisierte Druckermodell nach dieser Norm arbeitet. Sie erkennen das an den Worten »kompatibel zum Industriestandard« oder »Epson-kompatibel« in der Beschreibung des Druckers. Damit erwerben Sie die Gewißheit, daß die meisten professionellen Programme mit Ihrem Drucker auch funktionieren. Bei Commodore-Druckern brauchen Sie nicht nach dem »ESC/P-Stempel« zu suchen, denn alle Commodore-Drucker kennen nur einen einzigen Standard – ihren eigenen.

5. Der Preis

Die »Gretchenfrage« ist natürlich wie immer der Preis. Lohnt es sich, in einen guten Drucker zu investieren? Mit Sicherheit ja, lautet die deutlich ausgesprochene Antwort. Drucker kauft man am besten wie Kinderwäsche – lieber eine Nummer zu groß. Da Drucker einen sehr großen Anteil mechanischer Bauteile besitzen, sind billige Drucker in der Regel immer etwas weniger solide aufgebaut, als die etwas teureren Modelle. Trotzdem braucht man heute nicht mehr Unsummen auszugeben um einen soliden, leistungsfähigen Drucker zu erhalten. Man sollte aber überlegen, ob man nicht doch lieber gleich drei oder vier Hunderter drauflegt und dafür einen Drucker erhält, an dem man auch nach Monaten und Jahren wirklich Freude hat.

Druckerpalette

Wir stellen Ihnen nun eine bunte Palette verschiedener Drucker aller Preis- und Leistungsklassen vor, ganz besonders empfehlenswert sind dabei die drei 64'er Referenzdrucker (Preisklasse I bis III), denn sie haben in langen ausgiebigen Tests gezeigt, was in ihnen steckt. Aber auch die anderen Drucker können im Einzelfall für Sie interessant sein, denn jeder Drucker besitzt seine eigenen individuellen Vor- und Nachteile.

Die Problemlosen

Der MPS 801 (Bild 1) ist eine Weiterentwicklung des 1525 (baugleich mit Seikosha GP100VC, der vom GP500A abgelöst wurde). Alle Steuerzeichen und Sekundäradressen des C64 entsprechen denen des Druckers. Der Unterschied liegt im etwas modernisierten Gehäuse und einer anderen Druckmechanik (die des Seikosha GP500A). Das Farbband wurde gegenüber dem 1525-Drucker verkleinert und direkt auf dem Druckkopf in einer kleinen Kassette untergebracht. Durch die neue Mechanik ist der MPS 801 etwas schneller als sein Vorgänger geworden, er schafft jetzt 50 Zeichen pro Sekunde gegenüber 30 Zeichen pro Sekunde beim 1525.



Bild 1. MPS 801 von Commodore

```
ER BESITZT EINE 5X8 ZEICHENMATRIX
UND EINE DRUCKGESCHWINDIGKEIT VON 50
ZEICHEN PRO SEKUNDE
DER GP500A IST GRAFIKFAEHIG
DOPPELTE SCHRIFTBREITE
DURCH CHR$( 14 )
```

(Schriftbild verkleinert)

Bild 2. Schriftprobe vom MPS 801 und GP500A

Der MPS 801 kostet 298 Mark, bietet aber nur wenig Vorteile gegenüber dem 1525, der nur noch auf dem Gebrauchtmarkt erhältlich ist. Er eignet sich vor allem als Grafik-Drucker oder zum LISTen von Programmen. Für die Textverarbeitung gilt das gleiche wie für den Seikosha GP500A. Die Zeichendarstellung erreicht leider keine Briefqualität (Bild 2).

Ein Drucker für alle Gelegenheiten?

Rein äußerlich ist der MPS 803 (Bild 3) ein optisch ansprechendes, kompaktes Gerät, das sich auf altbekannte Art mit einem Kabel direkt an den C64 anschließen läßt. Neu sind zwei bisher bei Commodore nicht übliche Schalter auf der Geräterückseite. Diese sind mit LPI (Lines per Inch = Zeilenabstand) und Device (Geräteadresse) bezeichnet. Mit ihrer Hilfe wird zumindest das Einstellen dieser Werte problemlos.



Bild 3. Der MPS 803 - ein kompakter Drucker

```
DER MPS 803 KANN NICHT NUR GROSS-
sondern auch kleinschrift,
[illegible] REV AUS IN EINER ZEILE.
BREITSCHRIFT oder graphik-
ZEICHEN DARSTELLEN. ([illegible])
LEIDER FEHLEN DEUTSCHE ZEICHEN UND
Unterlaengen (payhgkj). Deshalb ergibt
sich kein harmonisches Schriftbild.
```


Bild 4. Das Schriftbild des MPS 803. Es fehlen Umlaute und Unterlängen

Der Papiereinzug des MPS 803 ist für Einzelblätter und Endlos-Rollenpapier gedacht, denn es fehlt jede Papierführung oder ein Traktorantrieb. Ein Traktor muß für etwa 100 Mark hinzugekauft werden. Diese Investition lohnt sich aber in jedem Fall, denn wenn man den Fehler begeht, das Papier nicht genau gerade einzulegen, so wandert es unweigerlich in die eine oder andere Richtung. Für eine Textverarbeitung ist der MPS 803 aber nicht nur wegen dieser Tatsache ungeeignet. Diese Anwendungen bleiben dem MPS 803 auch wegen seines Zeichensatzes vorenthalten. Der Zeichensatz ist auf den C64 abgestimmt und hat demzufolge keine Umlaute. Es besteht zwar die Möglichkeit, sich die Umlaute im Grafikmodus selbst zu definieren, leider wird diese Funktion nur von den wenigsten Textprogrammen unterstützt. Aber selbst wenn man sich damit abfinden könnte, das Aussehen der einzelnen Buchstaben, besonders der Buchstaben mit Unterlängen, kann nur als unharmonisch bezeichnet werden. Es gibt keine echten Unterlängen. Das heißt, alle Buchstaben mit Unterlängen werden einfach angehoben und schweben erhaben über dem Rest des Wortes. Zusammen mit dem ohnehin nicht überzeugenden Druckbild (Bild 4) wird wohl kaum jemand auf die Idee kommen, Briefe mit diesem Schriftbild zu schreiben. Der MPS 803 hat aber auch seine positive Seite - er ist grafikfähig. Dabei ist dieses Wort allerdings zu relativieren, denn die Grafik des MPS 803 mit der eines Star NL 10 zu vergleichen hieße, Äpfel mit Birnen zu messen. Auch die Steuerbefehle und die Druckgeschwindigkeit reißen niemanden aus dem Sessel, denn Breit- und Reversschrift sind einfach zu wenig.

Mit einem Marktpreis von etwa 398 Mark, zuzüglich 100 Mark für einen Traktorantrieb, ist der MPS 803 für seine Leistungen eindeutig zu teuer. Bedenkt man, daß der etwas bessere MPS 801 inzwischen günstiger angeboten wird (Preis etwa 298 Mark), so dürfte sich der Preis wahrscheinlich nicht mehr allzulange halten.

Der Präsident 6313 C – das preiswerte Schwergewicht

Mit rund sieben Kilogramm stellt sich der Präsident 6313 C als ein Schwergewicht unter den Druckern vor. Aber ist er auch ein Meister seiner Klasse? Wir haben es getestet.

Der Präsident 6313 C (Bild 5) erweckt zunächst den Eindruck, als ob er aus einer anderen Welt käme. Tatsächlich ist dieser Eindruck gar nicht so falsch, denn er ist ein ostdeutsches Produkt, das versucht, mit dem westlichen Standard Schritt zu halten.

Zwar ist ansprechendes Design (er wirkt etwas klobig) nicht gerade seine Stärke, aber ein praktisches Gerät wie einen Drucker kauft man ja nicht nur als Schmuckstück für die Wohnung. Wer möglichst lange etwas von seiner Investition haben möchte, wird vielmehr Wert auf andere Attribute legen.



Bild 5. Der Präsident - robust und preiswert

```
Präsident 6313 C
Schönschrift (NLQ)
Normalschrift
Breit
Fettschrift
Doppeldruck
NLQ+Fettschrift
NLQ+Doppelschrift
Breit/Fett
Hoch und Tief
```

Bild 6. Schriften, Made in DDR - der Präsident

In dieser Hinsicht hat der Präsident 6313 C allerdings einiges zu bieten. Er besitzt ein Chassis aus zwei Millimeter starkem Stahlblech, das alle anderen ebenfalls äußerst robust ausgeführten Teile aufnimmt. Dieser Eindruck wird dann noch bestärkt, wenn man den Drucker das erste Mal hochhebt, denn im Vergleich zu fernöstlicher Konkurrenz mutet dieses Schwergewicht wie ein prähistorischer Dinosaurier an. Wer dabei nicht aufpaßt, findet übrigens das Unterteil des Druckers recht unsanft auf seinen Zehen wieder, denn das

Gehäuseoberteil ist erstaunlicherweise nur mit einem Scharnier befestigt. Bei diesem Gewicht ist das sicherlich kein Genuß.

Hat man das Auspacken und Aufstellen bis hierher ohne Schaden an Leib und Drucker überstanden, wird man neugierig darauf, den leicht zu öffnenden Deckel wie die Kühlerhaube eines Autos nach oben zu klappen und das Innenleben des Druckers zu inspizieren.

Der Motor für den Druckkopf-Transport ist in einem Druckgußgehäuse untergebracht, die Druckkopfführung besteht aus einem Vierkantprofil und auch sonst findet man Eisen soweit das Auge reicht.

Der Druckkopf scheint etwas überdimensioniert, was aber sicher nicht negativ zu bewerten, sondern eher Ausdruck östlichen Technologiestandards ist.

Die Farbbandkassette hat die Maße 12 x 13 Zentimeter und läßt sich vorbildlich einfach einlegen, respektive wechseln. Die Finger behalten dabei ihre ursprüngliche Farbe, was letztendlich auch der Kleidung zugute kommt, denn Druckerfarbe gehört bekanntlich zu den hartnäckigsten Flecken.

Die Einstellung der Anschlagstärke ist, wie man nun schon fast erwartet, mechanisch einwandfrei, aber nur mit schlanken Fingern problemlos zu verstellen.

Der Antrieb des Druckwerkes erfolgt mit einem Stahlseilzug, der sogar eine Spannvorrichtung in Form einer einfachen Feder aufweist.

Der Präsident 6313 C verarbeitet sowohl Endlos- als auch Rollenpapier und Einzelblätter, die allerdings nicht automatisch eingezogen werden. Letztere werden ähnlich einer Schreibmaschine von oben hinter der Gummiwalze eingeführt und dann ausgerichtet. Der notwendige Papierlöser ist unmittelbar am Drehkopf angebracht. Bei Endlos- oder Rollenpapier wird der Einführschlitz an der Rückseite des Präsident 6313 C benutzt. Sicher ungewöhnlich ist die Anordnung der Stachelwalzen auf einer Achse mit der Gummiwalze. Leider ist er so gestaltet, daß beim Abreißen des Papiers an der nicht besonders scharfen Abreißkante immer ein Teil des nächsten Blattes verlorengeht.

Geradezu fortschrittlich erscheinen dagegen die 36 Mikroschalter, die unübersehbar auf der Vorderseite des geöffneten Präsident 6313 C angebracht sind. Diese Fülle an Schaltern löst sofort emsiges Blättern im (ost)deutschen Handbuch aus. Dort findet man dann Erstaunliches: Der Präsident 6313 C ist kompatibel zum Epson-Standard, zum IBM-PC, zu Commodore-Computern, und auch zu Schneider-Computern soll er passen. Sollte hier etwa doch ein Meister aller Klassen des Arbeiter- und Bauernstaates stehen? Es sei an dieser Stelle vorweggenommen: Der Präsident 6313 C zeigt, daß nicht nur im fernen, sondern auch im nahen Osten gute Drucker gebaut werden. Das Schnittstellenproblem ist durch das schon vom Epson GX-80 bekannte Modulkonzept gelöst worden. An der Rückseite des Druckers wird ein Modul eingeschoben, das die gesamte Elektronik für die jeweilige Schnittstelle enthält. Doch zurück zu den Schaltern. Mit ihnen wird zunächst die Wahl unter den verschiedenen Modul-Betriebsarten getroffen. Damit erhalten die restlichen Schalter völlig unterschiedliche Bedeutungen. Die Wahl aus zehn internationalen Schriftsätzen ist möglich, Pica, Elite, komprimierte oder fette Schrift können hier fixiert werden (Bild 6). Auch die recht ansprechende NLQ-Schrift hat einen eigenen Schalter. Alle Auswahlmöglichkeiten sind auch mit dem CHR$-Befehl zu erreichen, außer Kraft zu setzen oder zu verändern. Der Präsident 6313 C ist mit 96 Zeichen pro Sekunde in der Normalschrift kein Meister der Geschwindigkeit, aber er nadelt zuverlässig seine Zeichen aufs Papier und auch vor der Grafik schreckt er nicht zurück. Im NLQ-Modus befindet er sich mit 23 Zeichen pro Sekunde in guter Gesellschaft. Die Nadeln hämmern dabei mit einer Kraft aufs Papier, die ausreichend für bis zu drei Durchschläge ist.

Der Drucker Präsident 6313 C ist ein gutes Beispiel dafür, daß preiswerte Computertechnik nicht immer nur aus dem fernen Amerika oder Japan kommen muß. Bedenkt man, unter welchen Schwierigkeiten dieser Drucker dort entstanden sein muß, denn in der DDR gehört schon ein simpler EPROM zu den knappen Gütern, kann man die Konstrukteure nur beglückwünschen. Mit einem Preis von 798 Mark ist er der derzeit preiswerteste Drucker mit NLQ-Fähigkeit, bei dem keinerlei Abstriche an der mechanischen Festigkeit gemacht wurden. Da der Präsident 6313 C nicht gerade als ausgesprochene Schönheit gelten kann, kommt es bei ihm viel mehr auf seine elektronischen und mechanischen Fähigkeiten an – und die stimmen rundum.

Zwei zum Preis von einem

Der DX 2100 stellt sich als ein in zweierlei Hinsicht kompatibler Drucker vor. Zum einen soll er den Epson FX-85, zum anderen aber, und das ist das Außergewöhnliche, auch den Farbdrucker JX-80 emulieren können. Mit anderen Worten, der DX 2100 (Bild 7) strengt sich an, zwei Drucker in sich zu vereinigen, allerdings mit der Absicht, beide in ihren Leistungen zu übertreffen. Erreicht wird diese Flexibilität durch einen sinnvoll konstruierten Farb-Umrüstsatz, der so einfach

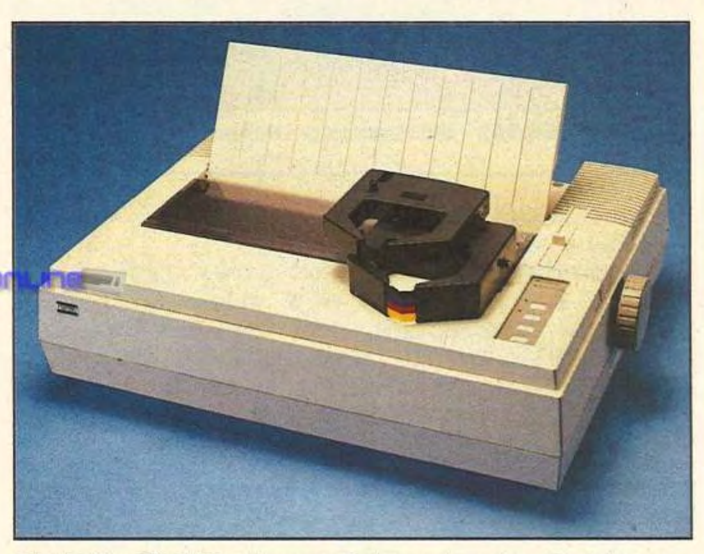


Bild 7. Der DX 2100, die neue Referenz in seiner Preisklasse

Fujitsu DX 2100
NLQ-Schönschrift
Breit
Normalschrift
Eliteschrift
Schmalschrift
Doppeldruck
Hoch- Tief-gestellt
Proportionalschrift

Bild 8. Trotz der enorm hohen Geschwindigkeit ein sehr gutes Schriftbild

Aa

Bild 9. Die Vergrößerung überzeugt von der guten Schriftqualitität

wie schnell einzubauen ist. Hat man diesen Umbausatz installiert, so genügt es, zwischen schwarzer und vierfarbiger Farbbandkassette, die sich garantiert ohne schmutzige Finger einlegen lassen, zu wählen, um den betreffenden Modus zu aktivieren. Eine Kerbe auf der Kassette signalisiert dem Drucker, ob er die Erlaubnis hat, das Farbband vor dem Kopf auf- und abzuschieben, um eine andere Farbe einzustellen. So wie die Farbfähigkeit als sinnvolle Ergänzung zu betrach-



ten ist, so bleiben die monochromen Fähigkeiten eines Druckers bei weitem die wichtigeren Beurteilungskriterien. Das Schriftbild ist hier entscheidend. Um aber ein gutes Schriftbild überhaupt und auf Dauer erzeugen zu können, bedarf es einer Menge Stahl. Beim DX 2100 hat dieser Stahl die Form von zwei soliden Gleitschienen angenommen, auf denen der vollkommen gekapselte 9-Nadel-Druckkopf mit der Präzision einer Schweizer Uhr gleitet. Eigentlich ist gleiten nicht ganz das richtige Wort, denn die Geschwindigkeiten, die der Druckkopf des DX 2100 entwickelt, sind beeindruckend. Mit einer Geschwindigkeit von angegebenen 220 Zeichen pro Sekunde (gemessen 210) schaffte er unseren Probetext in der bisher nicht erreichten Zeit von 1:30 Minuten. Der eigentliche Geschwindigkeitsvorteil gegenüber der gesamten Konkurrenz macht sich aber so richtig erst dann bemerkbar, wenn es an den Ausdruck in der exzellenten »Near Letter Quality«-Schrift (Bild 8 und 9) geht. Während die meisten Drucker die NLQ-Schrift nur im unidirektionalen Druck erzeugen können, druckt der DX 2100 weiterhin bidirektional. Das heißt, wenn der Druckkopf über eine Zeile hin- und wieder zurückgestrichen ist, ist die Zeile fertig. Das läßt sich natürlich auch mit Zahlen belegen; die Druckgeschwindigkeit beträgt in der NLQ-Schrift immerhin noch 44 Zeichen pro Sekunde und steht damit ebenfalls an der Spitze aller bisher getesteten Drucker.

Neben der exzellenten Schriftqualität haben sich die Konstrukteure des DX 2100 aber noch einige Besonderheiten einfallen lassen. So fällt es beispielsweise sehr angenehm auf, daß man das Endlospapier (einfach durch Druck auf eine Funktionstaste nach hinten transportieren), eingespannt lassen kann, wenn man mit Einzelblättern arbeitet. Sowohl das Endlospapier als auch die Einzelblätter werden automatisch per Tastendruck eingezogen und genauestens justiert. Der Papierantrieb ist dabei so geschickt hinter der Schreibwalze angebracht, daß das Papier auch tatsächlich an der als Abrißkante bezeichneten Stelle sauber abgetrennt werden kann.

Was wäre ein Drucker ohne seine Steuerbefehle? Beim DX 2100 hat man wohl eine der bemerkenswertesten Auswahlpaletten an ESC-Befehlen, denn sowohl die FX-85- als auch die JX-80-Farbsteuerbefehle sind vorhanden. Wer aber lieber die Schriften mit Schaltern oder Tasten einstellen möchte, braucht beim DX 2100 nicht lange zu suchen. Die vier Funktionstasten dienen, je nachdem, in welcher Reihenfolge, beziehungsweise Kombination sie gedrückt werden, dazu, 14 verschiedene Schriften beziehungsweise Funktionen aufzurufen oder zu löschen. Alle diese Funktionen sind auf die gängigsten Programme abgestimmt. Dank eines in dieser Form noch nie dagewesenen Steckkartenkonzepts kann praktisch jede beliebige Schnittstelle entwickelt werden und in Form einer Karte eingeschoben werden. Auch die Aufrüstung des internen Speichers von 2 KByte ist um 8 KByte oder 16 KByte mit zwei Handgriffen durch einfaches Einstecken der Karte möglich. Da die sonst so gerne versteckten DIL-Schalter beim DX 2100 auf der Speicherkarte angebracht sind, kann man sie zwar nicht unmittelbar, aber immerhin ohne große Umstände und ohne Werkzeug erreichen. Der Zugriff auf die DIL-Schalter ist aber wegen des weiter oben beschriebenen Bedienereinstellmodus nur in sehr seltenen Fällen notwendig.

Die Mischung aus perfektem Textdrucker und den Farbfähigkeiten des JX-80 scheint derzeit das ideale Konzept für Heim- und Personal Computer zu sein, die ja einerseits für Farbgrafiken, aber eben immer mehr zur Text- und Datenverarbeitung verwendet werden. Es sollte so sein, daß sich die Flexibilität der Computer auch in ihren Peripheriegeräten fortsetzt. Der DX 2100, der auch in einer IBM-Version erhältlich ist, ist ein ideales Beispiel dafür, wie diese Absicht in vertretbaren Preisregionen verwirklicht wurde.

Mit einem Listenpreis von 1932 Mark und einem Aufpreis von 456 Mark für die Farboption ist der DX 2100 ein Drucker, der, gemessen an seinen Leistungen, preiswert ist. Er ist unser Referenzdrucker der Preisklasse III.

Der Riesenzwerg

Gleich nach dem Auspacken des 120 D fällt das sehr kompakte Äußere auf. Mit 37,5 Zentimeter Breite und nur 23 Zentimeter Tiefe ein relativ kleiner Drucker, mit geringem Gewicht. Die schon fast üblichen Tasten ONLINE, FF und LF sind mit den optischen Kontrollen auch hier vorhanden. Auf die Funktionen dieser Tasten soll später eingegangen werden. Das Einlegen der Farbbandkassette verursacht zwar kein Kopfzerbrechen, dafür aber ganz schön schwarze Finger. Der Druckkopf macht einen etwas zierlichen Eindruck, ist aber seinen Aufgaben problemlos gewachsen. Von einem Zahnriemen getrieben, wird er auf zwei Lagern geführt. Davon ist das vordere eine Schiene und das hintere ein Winkel als Gegenlager. Damit wird die für die Schönschrift erforderliche Präzision erreicht, obwohl zwei Schienen die bessere Lösung wären. Als ein Drucker mit Epson- und IBM-Modus wird der 120 D (Bild 10) von einem deutschen Handbuch vorgestellt. Das Handbuch ist gut gegliedert und gibt, klar in der Darstellung, auf fast alle Fragen Antwort.

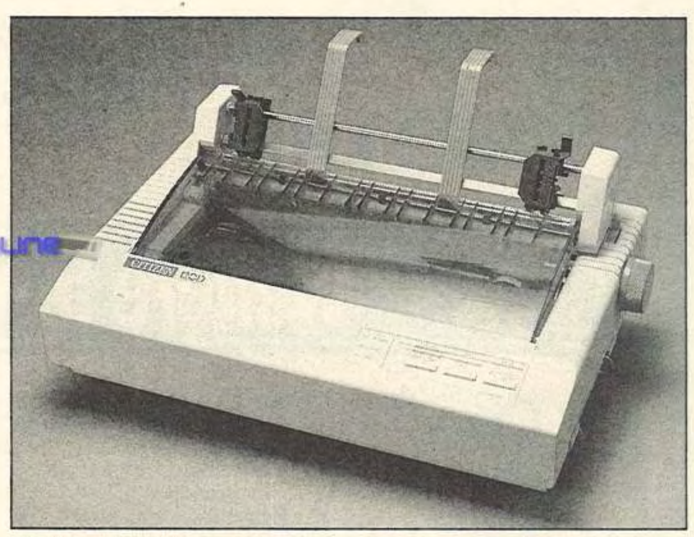


Bild 10. Der Citizen 120 D – neue Referenz der Preisklasse unter 1000 Mark

Normalschrift
Schönschrift
Doppeldruck
Eliteteilung
doppelte Höhe
Reverse
Breite
Breite/Höhe
Unterstreichen

Bild 11. Ein Teil der Möglichkeiten des 120 D

A a

Bild 12. Die NLQ-Schrift in fünffacher Vergrößerung

Auch der 120 D hat, wie die Geräte der MSP-Serie, einen sogenannten »Maintenance Self Test«, kurz MST, der über die eingebaute ROM-Version und das Datum der Programmierung Auskunft gibt. Anschließend werden Zeile für Zeile »H«s gedruckt, die der Überprüfung der Lage des Druckkopfes dienen. Leider wird die Stellung der DIP-Schalter im MSP

des 120 D nicht mehr angezeigt. Dies war angenehm bei dem MSP 10, ist aber beim 120 D auch nicht mehr unbedingt notwendig, da die DIP-Schalter von außen zugänglich sind. Den Entwicklern des 120 D ist zur Frage der Schnittstellen etwas Besonderes eingefallen. Das jeweils gewünschte Interface wird als Kassette eingeschoben. Diese Idee hat man perfektioniert und so lassen sich Schnittstellen bequem und ohne Werkzeug an der rechten Seite des 120 D einschieben. Die DIP-Schalter befinden sich in dieser Kassette und sind im eingeschobenen Zustand unter dem Druckkopfweg zugänglich. Eine gute Lösung, die sich auch bei anderen Druckern durchsetzen sollte. Wenn auch wegen der eben erwähnten Zugänglichkeit nicht mehr notwendig, so soll nicht verschwiegen werden, daß das Gehäuse des 120 D ohne jedes Werkzeug zu öffnen und zu schließen ist. Die Mechanik des 120 D ist gegenüber dem MSP 10 deutlich verbessert worden. Im Grundzustand ist der 120 D, ähnlich einer Schreibmaschine, nur für Einzelblätter geeignet. Diese lassen sich einfach und sicher einlegen. Wird die Verarbeitung von Endlospapier notwendig, so kann ein zum Lieferumfang gehörender Traktor mit einem Handgriff aufgesetzt werden.

Die Verriegelung der Stachelwalzen ist, gegenüber dem MSP 10, offensichtlich mit gutem Ergebnis überarbeitet worden und ist kein Grund zur Kritik mehr. Wegen des Aufsetzens und dem damit verbundenen längeren Transportweg des Papiers geht allerdings beim Entnehmen eines Dokuments ein Leerblatt verloren.

Eine gute Sache, die beim 120 D nochmals verbessert wurde, ist der vom MSP 10 bekannte Hex-Dump. Durch Drücken von LF und FF während des Einschaltens angewählt, wird man beim Ausdruck einer Textzeile überrascht seinen Augen nicht trauen. Nicht nur die HEX-Werte erscheinen auf dem Papier, sondern auf der rechten Seite des Blattes auch die dazu gehörenden ASCII-Zeichen. Ein Leistungsmerkmal, das den MSP 10 schon als anwenderfreundlich ausgewiesen hat.

Beim 120 D ist dieser Modus nochmals verbessert. Außer der Darstellung der ASCII-Zeichen werden auch »CR, SPACE, ESC« und andere Steuer-Befehle als Abkürzung dargestellt. Beim MSP 10 befand sich beispielsweise an der Stelle des »CR« ein Punkt. Beim 120 D erscheint das Kürzel »CR« so, daß beide Buchstaben übereinander stehen. Jeder, der schon einmal mit Handbüchern Hex-Werte in ASCII-Zeichen umgesetzt hat, wird diesen Komfort des 120 D schnell schätzen lernen.

Mit den Tasten ONLINE, LF und FF werden Hex-Dump, der Selbsttest, der Maintenance-Test und die Umschaltung in den NLQ-Modus vorgenommen. Das Einschalten des NLQ-Modus kann zeilenweise geschehen, das heißt, wird während eines Druckvorganges in den NLQ-Modus geschaltet, so wird die laufende Zeile ordnungsgemäß beendet und dann werden die folgenden Zeilen in Schönschrift aufs Papier gebracht. Ähnlich dem Epson FX-85 kann mit ONLINE und FF in einen Auswahlmodus geschaltet werden, der dann die Wahl verschiedener Darstellungsarten ermöglicht.

Papierbreiten von 76,2 mm bis hin zu 240 mm sind keine besondere Aufgabe für den 120 D. Dabei sind außer dem Original noch zwei Durchschläge möglich. Die Anschlagstärke wird an einem kleinen Hebel, seitlich rechts am Gehäuse, verändert. Keine Frage, auch Einzelblätter verarbeitet der 120 D mühelos.

Die hinlänglich bekannten, weil oft verwendeten ESC-Befehle dienen dem 120 D wie auch dem Vorbild zur Steuerung von Format-Anweisungen, Schriftartenumschaltung und der Ausführung sonstiger Steuerbefehle. Der 120 D verfügt im wesentlichen über den Befehlsumfang des FX-85, weshalb auch mit den wenigsten Programmen Probleme zu erwarten sind. Ein Befehl soll aber an dieser Stelle besonders hervorgehoben werden: CHR$(27) »h«. Mit diesem Befehl wird der 120 D veranlaßt, alle nachfolgenden Zeichen in doppelter Höhe zu Papier zu bringen (Bild 11).

Für 998 Mark wechselt der 120 D seinen Besitzer und zeichnet sich damit durch ein exzellentes Preis-/Leistungsverhältnis aus. Bedenkt man, daß sich der Marktpreis in der Regel noch etwas unterhalb dieses Listenpreises einpegelt, kann man von einer kleinen Sensation sprechen. Man erhält neben einem sehr guten Schriftbild in Near Letter Qualität (Bild 11 und 12) auch fast alle Funktionen des Epson FX-85 und sogar noch einen erweiterten Hex-Dump dazu. In der mechanischen Qualität zeigt der FX-85 durch seinen grundsoliden Aufbau allerdings, warum er fast 1000 Mark teurer ist. Trotzdem kann der Citizen 120 D ein Verkaufsschlager werden, denn ähnliche Leistungen wurden bislang nicht für so wenig Geld angeboten. Er ist unser Referenzdrucker der Preisklasse I.

Fast ideal – der NL-10

Mit dem NL-10 (Bild 13, 1145 Mark) hat Star versucht, alle Wünsche an einen guten Drucker zu erfüllen. Es hat allen Anschein, als ob es ihnen auch gelungen ist. Betrachten wir den NL-10 etwas genauer. Das Gehäuse ist beigefarben und paßt exzellent zum C 128, aber auch neben dem C 64 gefällt die Farbe. Wichtiger als die Farbe aber ist die Art, wie der NL-10 Papier verarbeitet. Während es beim SG-10 noch ein kleines Ärgernis war, daß durch den oben angebrachten Traktor jedesmal beim Einspannen ein Blatt verlorenging, hat der NL-10 den Traktor hinter der Schreibwalze im Gehäuse versenkt. Die Stachelwalzen lassen sich auf der ganzen Breite des Druckers verschieben, so daß auch schmale Etiketten problemlos eingespannt werden können. Links neben der Schreibwalze, die unter einer flachen Plastikabdeckung, die beim Anheben übrigens den Druckvorgang anhält, verborgen ist, befindet sich ein Multifunktionshebel. Seine Aufgabe

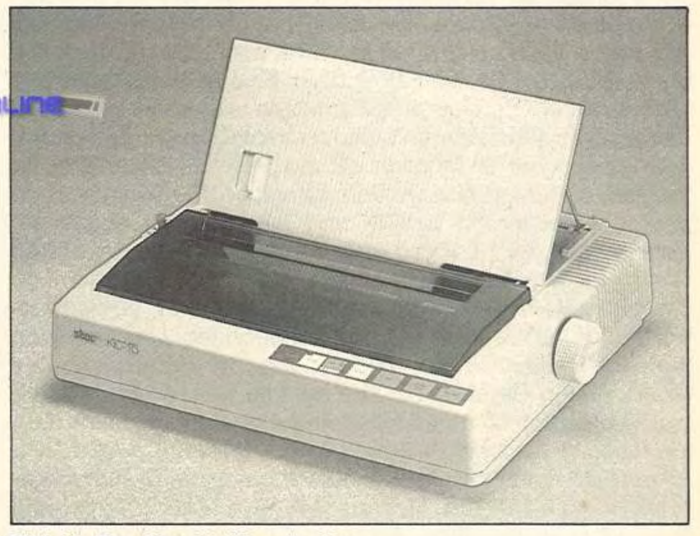


Bild 13. Der Star NL-10 - ein Star

Star NL-10
NLQ-Schönschrift
Normalschrift
Eliteschrift
Schmalschrift
Proportional
Breit
Fettschrift
Hervorgehoben

◄ Bild 14. Das Schriftbild des Star NL-10

Aa

Bild 15. Die NLQ-Schrift in fünffacher Vergrößerung

ist es, einzustellen, ob Einzel- oder Endlospapier eingespannt ist. Seine zweite Aufgabe ist es, den automatischen Einzelblatteinzug zu aktivieren. Ganz gleich mit welcher Papierart gedruckt werden soll, man legt einfach den Hebel nach hinten und das Papier wird automatisch eingezogen, zentriert und hinter den Andruckrollen fixiert. Der 9-Nadel-Druckkopf macht ebenso wie seine Führungsschienen einen sehr soliden und dauerhaften Eindruck. Links und rechts neben dem Druckkopf findet man etwas, das man normalerweise nur bei einer Schreibmaschine erwartet. Auf zwei Formstücken aus klarem Plastik sind hilfreiche Justierstriche angebracht, die besonders beim Ausfüllen von Formularen von Nutzen sind. Wer beim NL-10 eine Abrißkante für das Papier sucht, braucht dies nicht vergeblich zu tun, denn an der Abdeckung befindet sich eine Kante, die ihren Namen zu Recht trägt, denn sie ist messerscharf.

Der NL-10 arbeitet mit einer breiten Nylon-Farbbandkassette, die sich ohne Farbabdrücke auf den Fingern zu hinterlassen, einlegen läßt. Auch die DIL-Schalter gaben keinen Grund zur Kritik, denn sie befinden sich gut erreichbar auf der Gehäuserückseite.

Der NL-10 läßt sich wie ein Drucker der Spitzenklasse programmieren. Gut erreichbar auf der vorderen Gehäuseoberseite besitzt er fünf verschiedene Schalter und sieben Leuchtdioden. Mit den Schaltern lassen sich neben den üblichen Funktionen wie On Line, Zeilen- und Seitenvorschub die gewünschten Schriftarten bis hin zur exzellenten NLQ-Schrift (Bild 14 und 15) einstellen. Die fünfte Taste dient der Wahl der Fettschrift in Kombination mit der gewünschten Schriftgröße. Im Off-Line-Modus kann man per Tastendruck sogar die beiden Ränder einstellen. Der Druckkopf wird dabei von links oder rechts so lange in Mikroschritten bewegt, bis die gewünschte Position erreicht ist. Eine genaue Justierung auf besondere Formate und Formulare wird damit zum Spiel statt zum streßbeladenen Manöver. Der Clou der Tastenprogrammierung ist die Möglichkeit, die gewählte Einstellung zu fixieren. Natürlich lassen sich sämtliche Funktionen auch über die bekannten CHR$- und ESC-Befehle einstellen. Dabei wartet der NL-10 mit einer nützlichen und einer netten Raffinesse auf. Zunächst die nette. Hat man den Drucker fleißig mit verschiedenen Schriften programmiert, so flackern bei jedem Schriftwechsel die zugehörigen Leuchtdioden kurz auf. Wesentlich wichtiger aber ist der Befehlssatz des NL-10. Dieser Befehlssatz wird aber im wesentlichen durch das verwendete Schnittstellenmodul beeinflußt. Beim Kauf hat man die Auswahl zwischen einem Centronics- (ESC/P), einem IBM- und erfreulicherweise auch einem Commodore-Modul für C 64/C 128. Im Preis von 1145 Mark ist ein Modul nach Wahl einbegriffen. Mit dem Commodore-Modul, das einfach auf der Geräterückseite eingesteckt wird, ist der NL-10 direkt an den C 64, beziehungsweise C 128 anschließbar. Trotzdem bleibt die Möglichkeit, bei Bedarf ein Centronics- oder IBM-Modul (je 150 Mark) kaufen zu können. Der NL-10 ist wohl der einzige Drucker, bei dem die Programmierung des Commodore-Moduls so gut gelungen ist, daß man als C 64-, C 128-Besitzer getrost den Drucker mit entsprechendem Commodore-Modul kaufen kann. Trotz der Implementierung aller Befehle des MPS 803, einschließlich der Grafikbefehle, bleiben die Funktionen beispielsweise eines Epson LX-80 erhalten. Gleiches gilt für den Zeichensatz, hier hat man die freie Auswahl. Zum einen kann man den original Commodore-Zeichensatz oder den ASCII-Zeichensatz verwenden. Einer der DIL-Schalter ermöglicht es sogar, den Commodore-Zeichensatz an Stelle einiger Grafikzeichen um die deutschen Sonderzeichen und Umlaute zu bereichern. Das ganze haben wir mit verschiedenen Text- und Grafikprogrammen getestet. So druckt der NL-10 bespielsweise problemlos mit Vizawrite 64 und Textomat plus sowie ProText zusammen. Jedes erstellte Schriftstück hat alle Umlaute plus fast alle Commodore-Grafikzeichen. Dabei war es in keinem Fall notwendig, auch nur einen einzigen Befehl vorher an den Drucker zu übermitteln – alles funktioniert auf Anhieb! Bei den Grafikprogrammen war das Ergebnis sogar noch besser. Sowohl mit dem Copy-Befehl von Simons Basic (HiRes-Hardcopy), als auch mit Print Shop läßt sich ohne Anpassungsprobleme sofort drucken. Wohlgemerkt, das alles funktioniert, ohne auch nur einen Befehl aus dem übrigens sehr guten deutschen Handbuch wissen zu müssen. Bei diesen Programmen verwendet der NL-10 die 7-Nadel-Grafik des MPS 803. So richtig zum Zug kommt er aber erst, wenn man seine LX-80-Fähigkeiten verwendet. Dann ist er in der Lage, Grafiken mit bis zu vierfacher Dichte (wie der Fujitsu DX 2100) zu drucken. Mit dieser Grafik funktionieren übrigens alle Hardcopy-Routinen für Epson-Drucker. Nicht unerwähnt bleiben soll auch der Hexdump-Modus, bei dem der NL-10 sowohl Hex-Zahlen als auch deren Dezimalwerte in Klarschrift darstellt.

Auch für Programmierer

Wer lieber seine eigenen Buchstaben entwerfen will, hat dazu ausreichend Gelegenheit, denn sowohl die Standard-, als auch die NLQ-Zeichen sind vor keiner Veränderung sicher. Seine exzellenten Schriftqualitäten (Bild 8) prädestinieren den NL-10 zur Textausgabe. Trotzdem besitzt er auch Fähigkeiten, die das Herz jedes Programmierers höher schlagen lassen. Der NL-10 druckt jedes Commodore-Zeichen korrekt, so wie es auch auf dem Bildschirm zu sehen ist. Mit den Cursor- und Farbsteuerzeichen macht er aber etwas ganz besonderes. Ein Listing, mit dem NL-10 erzeugt, sieht automatisch ähnlich aus, wie unsere Checksummer-Listings, denn alle wenig aussagekräftigen Steuerzeichen werden in Klarschrift übersetzt. Die manchmal schon nervtötende Suche in verschiedenen Vergleichstabellen findet mit dem NL-10 endgültig ihr Ende. Er ist unser Referenzdrucker Preisklasse II. (aw)




TEXT-
MANAGER



Basic-Kenntnisse gehören heute zur Allgemeinbildung! Ein Commodore-Besitzer muß einfach Basic können, um mit seinem Computer vernünftig zu arbeiten. Denn die meisten Programme sind in Basic geschrieben. Sie werden staunen, wie schnell Sie ein funktionierendes Basic-Programm geschrieben haben! Hier zeigen wir allen Neulingen am Computer, wie man programmiert.

Alle Anfänge werden begleitet von guten Vorsätzen! Am Anfang dieses Basic-Kurses möchte ich keine Ausnahme machen und sogar soweit gehen, Ihnen meine guten Vorsätze offen darzulegen.

1. Ich habe den Kurs so geschrieben, daß Sie nur wenige Voraussetzungen mitbringen müssen, um ihn zu verstehen. Ich gehe davon aus, daß Sie die ersten Kapitel des Handbuches gelesen und angewendet haben, welches Ihnen beim Kauf des Computers mitgeliefert worden ist.
2. Ich möchte gern ohne Fach-Chinesisch erklären. Wo ich ohne Fachausdrücke nicht auskommen kann – und das wird natürlich öfters vorkommen – werde ich sie erklären.
3. Der Kurs beginnt ganz langsam, mit vielen leichten Beispielen und Wiederholungen.
4. Der Kurs soll aber nicht langweilig sein. Er wird daher bald an Tempo zunehmen, im Vertrauen darauf, daß Sie sehr bald keine reinen Anfänger mehr sein werden.
5. Ich gehe davon aus, daß Sie sich nicht ein rein theoretisches Basic aneignen, sondern programmieren lernen wollen. Das bedeutet aber, daß ich Ihnen auch Bedienungshinweise und Erklärungen der Arbeitsweise des Computers geben muß – so wenig wie möglich und so viel wie nötig.
6. Der Kurs ist zwar speziell auf den C 64 zugeschnitten. Die meisten Angaben treffen allerdings auch auf den C 16 und den VC 20 zu.
7. Der letzte Vorsatz ist eigentlich eine Aufforderung an Sie, liebe Leser. Wenn Sie etwas nicht verstanden oder spezielle Fragen haben, dann schreiben Sie mir bitte über den Verlag. Ich werde die Fragen entweder direkt beantworten oder aber die Fragen sammeln und Ihnen im 64'er eine eigene Frage-Antwort-Folge widmen.

Bevor wir anfangen, will ich noch schnell die Methode erwähnen, mit der Sie am besten den Kurs verfolgen und den Stoff nachvollziehen können.

Ich möchte diese Vorgehensweise »SIMULTAN-METHODE« (Bild 1) nennen, was nichts anderes bedeutet, als das Heft mit Basic-Kurs vor dem Computer sitzend zu lesen und alle Angaben, die mit dem Zeichen ⟶ gekennzeichnet sind, sofort einzutippen und auszuprobieren.

Ich bin mir bewußt, daß ich dadurch eine Arbeitsweise begünstige, die von vielen Programmierspezialisten generell verteufelt und ganz besonders der Sprache Basic vorgeworfen wird, nämlich »Hackerei«.

Auf die Folgen der Hackerei, wie und wann man ihre Nachteile vermeiden kann, werde ich noch oft zurückkommen. Aber dazu müssen Sie erst einige Dinge über Basic wissen. Bis dahin will ich ganz bewußt die positiven Seiten des Hackens in Anspruch nehmen.

So, jetzt habe ich keinen Grund mehr, weitere schöne Einleitungsphrasen zu dreschen. Jetzt soll's losgehen.

Voraussetzung für ein Programm - egal, in welcher Sprache es geschrieben ist - ist eine Einrichtung, dem Computer mitzuteilen, was er tun soll und eine andere Einrichtung, um zu sehen, was er tut beziehungsweise was er getan hat. Dazu dient als allererstes die Tastatur und der Bildschirm.

Sobald wir den Computer einschalten, ist das Eingabegerät Tastatur direkt mit dem Ausgabegerät Bildschirm verbunden. Im Handbuch Ihres Computers haben Sie sicher schon nachgelesen, wie die Tastatur funktioniert und wie damit Zeichen auf den Bildschirm gezaubert werden. Ich will das nicht im Detail wiederholen, aber ein paar Gedanken ist das Thema »Tastatur«schon wert.

Zeichen-, Steuer- und Funktionstasten

Ich teile die Tasten in drei verschiedene Typen ein:
- Zeichentasten
- Steuertasten
- Funktionstasten

ZEICHENTASTEN erzeugen, wie ihr Name sagt, Zeichen. Dazu gehören Buchstaben und Zahlen, mathematische Zeichen (Addition, Multiplikation, runde Klammern etc.), Symbole (Dollar, Gänsefuß, Pfeile und so weiter) und grafische Zeichen. Die grafischen Zeichen stehen nicht oben auf den Tasten, sondern auf ihrer Vorderseite.

Jede Zeichentaste bietet mehrere Zeichen und Symbole zur Auswahl. Um die verschiedenen Zeichen einer Taste auszuwählen und auf den Bildschirm zu bringen, verwenden wir spezielle Tasten wie SHIFT, CTRL und so weiter Diese Tasten nenne ich STEUERTASTEN.

Sie steuern auf sehr direkte Art und Weise alle Vorgänge auf dem Bildschirm. Aber sie erlauben auch, innerhalb von Programmen deren Abläufe zu steuern und zu verändern.

Ich halte diese Tasten für wichtig genug, um trotz ihrer Behandlung im Handbuch von Commodore etwas näher auf sie einzugehen.

Bevor ich das mache, will ich der Vollständigkeit halber auch die dritte der von mir genannten Gattung, nämlich die Funktionstasten erwähnen.

Mit FUNKTIONSTASTEN werden die 4 senkrecht untereinander angeordneten Tasten f1 bis f7 rechts außen bezeichnet. Sie sind universell für alle möglichen Funktionen einsetzbar, können diese Funktionen aber erst dann erfüllen, wenn sie entsprechend programmiert worden sind. Aus diesem Grund werde ich sie erst später behandeln und ich lasse sie vorläufig links - Verzeihung, natürlich rechts - liegen.

Zurück zu den Steuertasten, von denen ich oben sagte, daß sie im direkten Gebrauch Dinge steuern, die sich auf dem Bildschirm abspielen.

Ich meine damit zum Beispiel die CTRL-Taste, mit der die acht verschiedenen Zeichenfarben schwarz bis gelb eingestellt werden können. Ich meine damit auch die SHIFT-Taste, welche den rechten Teil der Zeichen und grafischen Symbole umschaltet. Dazu gehört auch die Taste links unten, die das Firmenzeichen von Commodore trägt und welche die linken Symbole umschaltet.

Dazu gehören schließlich und letztlich die Cursor-Steuertasten CRSR, die INST/DEL-Taste und die CLR/HOME-Taste.

Diese letztgenannten Tasten sind eng mit der Wirkungsweise des sogenannten EDITORS verbunden.

Editor ist ein englisches Wort und heißt soviel wie Redakteur. In unserem Fall ist der Editor ein im Computer fest eingebautes Programm, welches wie ein Redakteur dafür sorgt, daß auf dem Bildschirm alle Zeichen und Symbole in den entsprechenden Farben auf den richtigen Platz kommen.

Ein recht sichtbares Hilfsmittel, dem Editor unsere Wünsche und Vorstellungen mitzuteilen ist der CURSOR.

Die Bedeutung und seine Handhabung haben Sie ja sicher schon dem Handbuch entnommen. Das Wort »Cursor« kommt nicht vom englischen »curse« = verfluchen (obwohl das Verhalten des Cursors oft dazu verleitet), sondern aus dem lateinischen Wortschatz, wo es »Läufer« bedeutet.

Der Cursor läuft also über den Bildschirm, gesteuert vom Editor. Wir können seinen Lauf auch steuern, und zwar mit den beiden Cursor-Steuertasten (ach so!).

Eine andere Steuerung des Cursors bietet uns die CLR/HOME-Taste rechts oben. Allein gedrückt - dann gilt HOME - bringt sie den Cursor »nach Hause« nämlich in die linke obere Ecke des Bildschirms. Wird die Taste mit SHIFT umgeschaltet, bedeutet sie CLR - das heißt CLEAR, auf deutsch FREIMACHEN oder LÖSCHEN. Jetzt wird nämlich der Cursor nach links oben gebracht und zusätzlich der Bildschirm gelöscht.

Probieren Sie bitte diese Cursor-Bewegungen aus.

→ Am besten fahren Sie den Cursor in die Mitte des Bildschirms, tippen ein paar beliebige Buchstaben ein und drücken dann die HOME-Taste.

→ Wiederholen Sie das Ganze und verwenden Sie dann die CLR-Taste.

Der Editor bietet uns noch einen anderen sehr lobenswerten Service. Er berücksichtigt nämlich, daß wir alle sehr menschliche Menschen sind, die nicht nur viele Fehler machen, sondern auch immer wieder ihre Meinung ändern. Der Editor erlaubt uns, Fehler zu korrigieren oder bereits getippte Zeichenfolgen abzuändern.

→ Wir können mit dem Editor ungestraft über vorhandene Zeichen fahren, ihn nach Lust und Laune anhalten und dort, wo er gerade blinkt, ein neues Zeichen eintippen.

Ich nenne das »überschreiben«. Wo wir ein Überschreiben nicht anwenden können, hilft uns die INST/DEL-Taste (rechts oben) weiter.

INST ist die Abkürzung für INSERT, das heißt soviel wie EINFÜGEN.

DEL bedeutet DELETE, und das heißt ENTFERNEN oder AUSLÖSCHEN.

Im Handbuch steht nur eine kurze Beschreibung der Wirkung dieser Taste, die aber nicht unbedingt ausreichend ist. Schon erste Versuche zeigen, daß die Taste ihre Tücken hat. Ich bin dafür, DEL und INST einfach auszuprobieren.

→ Schalten Sie den Computer aus und dann wieder ein.

→ Fahren Sie mit dem Cursor auf das A von READY. Er steht jetzt an der dritten Stelle vom linken Bildrand.

→ Drücken Sie die DEL-Taste. Das E ist verschwunden, es steht nur noch RADY da, der Cursor blinkt immer noch auf dem A, er ist aber jetzt an die zweite Stelle vom linken Bildrand gerückt.

→ Ein zweiter Druck auf die DEL-Taste reduziert das Wort ADY, und der Cursor steht jetzt auf der ersten Stelle.

Die DEL-Taste löscht also das links neben dem Cursor stehende Zeichen und verschiebt den Cursor mitsamt dem ganzen rechten Schwanz eine Stelle nach links. Was passiert aber, wenn der Cursor am linken Bildrand angestoßen ist?

→ Ein dritter Druck auf die DEL-Taste bringt den Cursor an den rechten Bildrand eine Zeile darüber, nur da gibt es gerade nichts zu löschen. Seinen ADY-Schwanz läßt er am Anfang der unteren Zeile zurück.

→ Lassen Sie DEL-Taste gedrückt. Der Cursor läuft kontinuierlich weiter nach links, löscht alles, was ihm in den Weg kommt - solange, bis er »zu Hause« links oben angekommen ist.

Der geSHIFTete Teil dieser Taste, nämlich INST ist ebenso trickreich.

→ Schalten Sie den Computer aus und wieder ein.

→ Fahren Sie mit dem Cursor wieder auf das A von READY.

→ Drücken Sie die INST-Taste (SHIFT + INST/DEL). Der Cursor bleibt auf seiner dritten Stelle vom linken Bildrand stehen. Aber das Zeichen, auf dem er blinkt und



Bild 1. So funktioniert die Simultan-Methode

alles, was rechts von ihm steht, wird eine Stelle nach rechts geschoben. Wo der Cursor blinkt, entsteht eine freie Stelle, auf die direkt ein neues Zeichen geschrieben werden kann.

→ Durch mehrfaches Drücken der INST-Taste werden mehrere Stellen freigeschoben, in die mehrere Zeichen hintereinander eingefügt werden können.

Dieses Freimachen beziehungsweise Einfügen geht nicht in beliebiger Länge, da der Editor – wie ein echter Redakteur auch – gewissen Einschränkungen unterliegt. Ich würde natürlich am liebsten jetzt weiter über den Editor mit allen seinen Regeln und Einschränkungen dozieren. Und irgendwann muß ich es auch noch tun, denn beim Experimentieren werden Sie sicher noch darauf stoßen und wissen dann vielleicht nicht weiter.

Damit Sie aber nicht ungeduldig werden, möchte ich auf weitere Feinheiten vorläufig verzichten und das bisher Gelernte für die ersten Schritte in Basic anwenden.

FAZIT
Die Zeichen- und Steuertasten erzeugen eine direkte Wirkung auf dem Bildschirm. Der Bildschirm wird von einem eingebauten Programm verwaltet – dem Editor. Er ermöglicht beliebige Änderungen und Korrekturen von bereits eingegebenen Texten und Zahlen.

Computer bedeutet Rechenmaschine

Dies weist uns den Weg für das weitere Vorgehen.

Wenn jeder simple Taschencomputer rechnen kann, dann müssen es die Commodore-Computer auch können. Dazu will ich Ihnen noch schnell die Symbole der mathematischen Funktionen aufschreiben.

FUNKTION	SYMBOL	BEISPIEL
Addition	+	3+2
Subtraktion	-	3-2
Multiplikation	*	3*2
Division	/	3/2
Potenz	↑	3↑2

Die unterste Funktion ist wahrscheinlich die für Sie ungewohnteste. Das Beispiel wird im Klartext als »drei hoch zwei« oder als »drei Quadrat« gelesen.

Beim Taschenrechner wird normalerweise eine Rechnung so eingegeben:

→ `3 + 2 =`

Aber beim C 64 rührt sich da gar nichts, außer daß diese Zeile auf dem Bildschirm steht.

Und warum passiert nichts??

Überlegen Sie – alles, was wir gemacht haben – war, per Tastendruck Zahlen und Symbole einzutippen. Wir haben lediglich den Editor auf Trab gehalten, den Computer selbst haben wir damit nicht aufgeweckt.

Dieser Faulpelz schläft nämlich so lange, bis er einen Auftrag bekommt, etwas auszuführen. Für diesen Tritt, der ihn hinter dem Ofen hervorscheucht, brauchen wir eine der vorher zwar erwähnten, aber noch nicht eingesetzten Steuertasten zur Programmsteuerung.

Sie sitzt am rechten Rand der Tastatur und ist mit RETURN gekennzeichnet.

Schreiben wir also die Zeile von oben nocheinmal. Dieses Mal bitte ich Sie aber, nicht die Eingabe-Technik der Taschencomputer zu verwenden, sondern mir eine »lesbare« Schreibweise nachzumachen – Sie werden gleich sehen, warum.

Schließen Sie die Eingabe mit der RETURN-Taste ab.

Das sieht dann so aus:

→ `SUMME = 3 + 2`

→ `Drücken Sie die RETURN-Taste`

Aber wir sehen ja noch immer nichts!

Aller Anfang ist schwer. Wir müssen berücksichtigen, daß der Computer nichts, aber auch gar nichts von allein macht.

Für alles braucht er eine Anweisung. Nun, oben haben wir ihm die Anweisung gegeben, die Summe aus 3 plus 2 zu bilden. Das hat er auch gemacht, glauben Sie es mir. Aber wir haben ihm nicht gesagt, was er mit der Summe machen soll, und so hat er sie sich einfach nur gemerkt.

Wir wollen sie natürlich auf dem Bildschirm sehen. Dazu müssen wir dem Computer eine spezielle Anweisung geben.

Dieser Befehl, irgend etwas auf dem Bildschirm auszudrucken, lautet in Basic

PRINT.

Nach PRINT, was auf deutsch »drucken« heißt, geben wir dem Computer an, was er ausdrucken soll – in unserem Beispiel die Summe.

→ `PRINT SUMME`

→ `Drücken Sie die RETURN-Taste.`

Heureka! Endlich sind wir weitergekommen. Auf dem Bildschirm steht, vom Editor in die nächste Zeile gebracht, die Zahl, welche der Computer für die SUMME ausgerechnet hat.

Basic-Befehl Nr. 1 PRINT
druckt alles, was hinter dem Wort »PRINT« steht, auf den Bildschirm aus.

Ich schlage vor, daß Sie mit dem ersten Wort der Sprache Basic, das Sie nun gelernt haben, noch ein bißchen üben.

Bilden Sie das Produkt von 15 mal 14. Das geht doch einfach, oder?

→ `PRODUKT = 15 * 14`

→ `nicht vergessen RETURN zu drücken`

(Sie wissen doch, mit RETURN geben wir dem Computer die Anweisung, die getippte Zeile auszuführen)

→ `PRINT PRODUKT`

→ `RETURN drücken`

Wenn Sie gut Kopfrechnen können, dürfen Sie das Resultat nachprüfen. Oder vielleicht nehmen Sie einen Taschenrechner! Ich wette, irgend ein Leser sagt jetzt: »Was soll der Quatsch. Man kann doch die Rechnung 15*14 direkt mit PRINT ausführen. Der Umweg über SUMME= oder PRODUKT= ist doch völlig unnötig«.

Tja, liebe Leser, das stimmt in der Tat. Diese Kurzform ist erlaubt, und ich will Sie Ihnen gleich vorführen, vielleicht mit der Division. Wir wollen 3 durch 2 teilen. Geben Sie dazu ein:

→ `PRINT 3/2`

→ `RETURN drücken`

Wir erhalten sofort den Wert 1,5 – aber halt!! Er ist 1.5, also nicht mit Komma, sondern mit Dezimalpunkt geschrieben. Das ist die amerikanische Schreibweise, an die Sie sich

gewöhnen müssen als Preis dafür, daß Sie mit amerikanischen Computern arbeiten.

Also – der Befehl PRINT führt eine nachfolgende Rechnung direkt aus und druckt das Resultat auf den Bildschirm. Ist demnach meine oben genannte Methode, die Rechnung in zwei Schritten auszuführen, tatsächlich Quatsch?

Meine Antwort ist ein klares Nein. Beide Methoden haben ihre Berechtigung, und ich will Ihnen auch den Unterschied zeigen.

Mit der ersten Methode (wir haben sie bei SUMME und PRODUKT angewendet) erhält der Computer zuerst den Auftrag, eine Rechnung durchzuführen. Das Resultat merkt er sich unter dem Stichwort SUMME beziehungsweise PRODUKT. Haben Sie es gelesen? Er merkt sich das Resultat und legt es in einem Speicher (Bild 2) ab. Das bedeutet, daß wir es mit PRINT so oft wir wollen, auf den Bildschirm drucken können. Geben Sie jetzt gleich nochmal ein:

→ `PRINT SUMME`
→ `RETURN drücken`

Und siehe da, das Resultat von vorhin erscheint wieder. Dasselbe geht mit dem Stichwort PRODUKT.

Sie brauchen übrigens die Zeile nicht neu einzugeben. Vergessen Sie den Service des Editors nicht!

→ Fahren Sie mit dem Cursor auf die letzte PRINT-Zeile und überschreiben Sie das Wort SUMME mit dem Wort PRODUKT, dann muß nur noch die RETURN-Taste gedrückt werden.

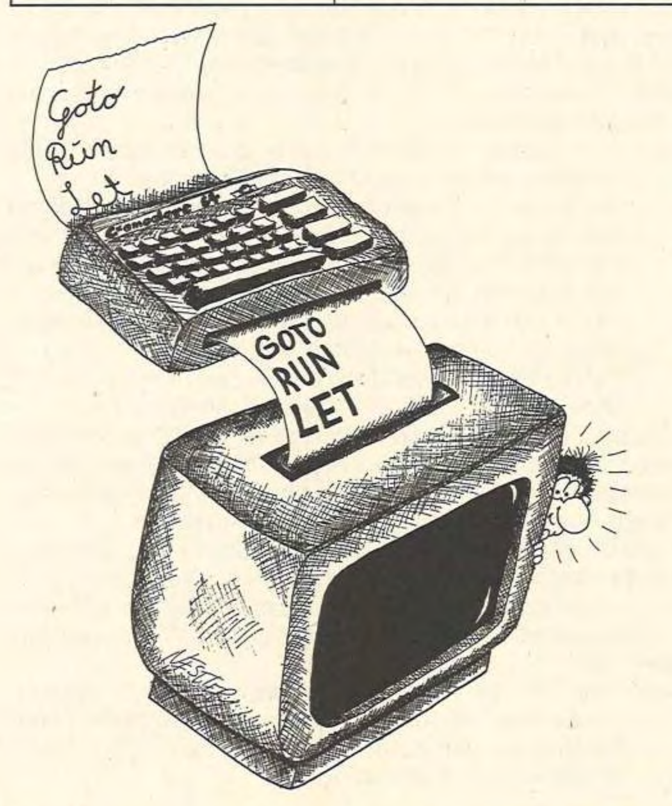
Und noch einmal siehe da, wir erhalten das Resultat der Multiplikation wieder.

So, und was ist mit der Division?

Leider, leider! Da geht nichts mehr. Wir haben kein Stichwort für das Resultat vorgegeben, und der Computer hat sich auch nichts gemerkt.

Das ist also der Unterschied:

• PRINT 3 + 2	• SUMME = 3 + 2 • PRINT SUMME
Das Resultat der Rechnung wird ausgedruckt	Das Resultat der Rechnung wird unter dem Stichwort SUMME gespeichert und dann ausgedruckt



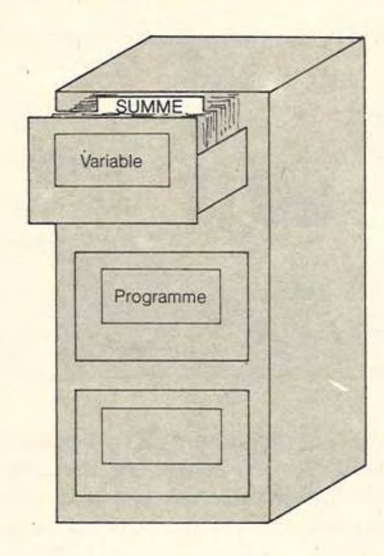


Bild 2. Speicher

Die linke Anwendung des Befehls PRINT ist einfach, und Sie können sie immer verwenden, um auf Art eines Taschenrechners schnell einmal irgendwelche Rechnungen zu machen. Aufgezeichnet wird das Resultat allerdings nur auf dem Bildschirm. Im Computer selbst bleibt nichts gespeichert.

Die rechte Art ist schwieriger, aber interessanter. Es ist eigentlich erstaunlich, daß wir dem Computer die Anweisung geben, sich eine Zahl unter einem Namen oder – wie ich es vorhin genannt habe – unter einem Stichwort zu merken, ohne einen Basic-Befehl geben zu müssen.

Es gibt tatsächlich einen Befehl dafür, er heißt:

LET

Die Anweisung mit dem Stichwort »SUMME« sieht unter Zuhilfenahme des Befehls LET so aus:

`LET SUMME = 3 + 2`

In unseren Sprachgebrauch übersetzt heißt das ungefähr: »Laß die Summe gleich dem Resultat von 3 plus 2 sein«. In der Computersprache nennt man das eine ZUWEISUNG.

Wir können demnach beliebige Stichwörter hernehmen und ihnen mit dem LET-Befehl Zahlenwerte zuweisen, der Computer merkt sie sich alle.

Wie würden Sie zum Beispiel dem Stichwort A (kürzer kann es nicht sein) den Wert 375 zuweisen, dann dem Stichwort X den Wert 15 und schließlich die Division von vorhin, die der Computer sich gemerkt hat, jetzt dauerhaft durchführen ? Versuchen Sie es:

So muß es aussehen:

→ `Löschen Sie den Bildschirm (CLR/Home+SHIFT)`
→ `LET A = 375 (RETURN drücken)`
→ `LET X = 15 (RETURN drücken)`
→ `LET QUOTIENT = 3/2 (RETURN)`

Jetzt sind im Computer die zwei Werte unter den Stichworten A, X und das Ergebnis der Division unter dem Namen QUOTIENT gespeichert.

Mit PRINT-Befehlen können wir alle drei auf den Bildschirm bringen:

→ `PRINT A (RETURN drücken)`
→ `PRINT X (RETURN drücken)`
→ `PRINT QUOTIENT (RETURN drücken)`

SUMME, PRODUKT, A, X und QUOTIENT, die wir bisher »Stichwörter« oder »Namen« genannt und welchen wir mit dem LET-Befehl einen Wert zugewiesen haben, werden mit einem eigenen Fachausdruck bezeichnet.

Sie heißen Variable (siehe nächste Seite). Ich werde später noch mehr über sie berichten. Heute bitte ich Sie lediglich, sich den Namen zu merken – manchmal werde ich vielleicht auch noch Stichwort sagen.

Wozu brauchen wir den Befehl »LET« ? Am Anfang haben wir ohne ihn den Stichwörtern Resultate zugewiesen!



In Basic kann in der Tat der Befehl LET weggelassen werden. Wichtig ist lediglich, daß zuerst die Variable dasteht, gefolgt von dem Gleichheitszeichen (=) und dahinter der Wert (oder der durch die Rechnung auszurechnende Wert).

Basic-Befehl NR.2 LET
weist einer Variablen einen Wert zu. Er kann auch weggelassen werden. Es genügt der Name der Variablen, ein Gleichheitszeichen und der zugewiesene Wert.

Interessant ist übrigens, daß der Computer sich ein- und dasselbe Stichwort nur einmal merkt. Wenn Sie zuerst dem Stichwort QUOTIENT den Wert 3/2 zuweisen und gleich danach den Wert 15/3, erhalten Sie mit dem PRINT-Befehl nicht die Zahl 1.5, sondern 5.

Das erste Resultat hat der Computer in seinem Speicher durch das zweite überschrieben. Probieren Sie es aus:

```
→ PRINT QUOTIENT (RETURN drücken)
→ LET QUOTIENT = 15/3 (RETURN drücken)
→ PRINT QUOTIENT (RETURN drücken)
```

Zuerst kommt der alte Quotient von vorhin, danach der neue Wert.

FAZIT

1. Nicht nur wir können Zeichen auf den Bildschirm bringen - der Computer selbst kann das auch. Der Befehl PRINT druckt alles, was dahinter steht, auch Ergebnisse von Rechenanweisungen, auf den Bildschirm.
2. Unter einem Stichwort (Variable genannt) kann sich der Computer Zahlenwerte merken, die beliebig oft mit dem PRINT-Befehl ausgedruckt werden können. Voraussetzung ist lediglich, daß mit dem LET-Befehl dem Stichwort der Zahlenwert zugewiesen wird. Das Wort »LET« selbst kann auch weggelassen werden.

PRINT ist ein sehr vielseitiger Befehl

Ist Ihnen das viele PRINT-Tippen, so wie bei den Beispielen oben, auch so lästig wie mir?

Diesen Dauerversuch mit unzähligen PRINT-Befehlen können Sie vereinfachen. Alle Commodore-Computer sind nämlich Meister der Abkürzung. So drastisch, wie wir den Befehl LET durch Weglassen abgekürzt haben, dürfen wir den Befehl PRINT nicht behandeln. Aber immerhin: Er wird mit dem Fragezeichen abgekürzt.

Das Befehlswort »PRINT« kann durch das »?« ersetzt werden

Statt PRINT 123
dürfen Sie schreiben ? 123

Jetzt geht der Versuch, alle auf den Tasten vorkommenden Zeichen und Symbole über den Befehl PRINT auszudrucken, natürlich viel schneller. Machen Sie bitte Gebrauch von dieser Abkürzung, wann immer Sie wollen.

Ich werde allerdings stets das ganze Befehlswort verwenden, damit der Text gut lesbar bleibt.

Die Sprache Basic gewährt uns noch eine Marscherleichterung. Es ist nämlich erlaubt, mehrere auszudruckende Werte hinter einen einzigen PRINT-Befehl zu setzen.

Zuerst will ich Ihnen zeigen, wie Sie es nicht machen dürfen - probieren geht über studieren.

Haben Sie die alten Stichwörter noch im Computer?

Wenn ja, dann brauchen Sie die folgenden Zuweisungen nicht nochmal eingeben. Ich wiederhole sie für alle Leser, die inzwischen ihren Computer ausgeschaltet und damit sein Gedächtnis gelöscht haben.

```
→ A = 375 (RETURN drücken)
→ X = 15 (RETURN drücken)
→ QUOTIENT = 15/3 (RETURN drücken)
```

Sie sehen, ich habe LET weggelassen.

So, die Werte haben ihr Stichwort beziehungsweise den Kennbuchstaben, unter dem sie der Computer in seinem Speicher wiederfindet. Jetzt wollen wir sie mit einem einzigen PRINT-Befehl ausdrucken:

```
→ PRINT A X QUOTIENT (RETURN drücken)
```

Der Computer druckt eine Null aus. Wie gesagt, so geht's nicht.

Wenn ich Ihnen sage, warum eine Null erscheint, werden Sie gleich auf die Lösung des Problems kommen. Was passiert da?

Für den Computer ist eine Leerstelle (zwischen den Variablen) genau so ein Zeichen wie jeder Buchstabe. Das heißt, daß er nach dem Drücken der RETURN-Taste in seinem Speicher nach einem Stichwort »A X QUOTIENT« sucht - und das findet er natürlich nicht.

Ihnen ist sicher klar, daß wir also die drei Stichwörter voneinander trennen müssen. Die Rechtschreibregeln von Basic erlauben keine Wörtertrennung mit Leerzeichen, sondern mit dem Semikolon »;«.

Verbessern Sie bitte (durch einfaches Einfügen von zwei Semikolons) die letzte Zeile oben:

```
→ PRINT A;B;QUOTIENT (RETURN drücken)
```

Jetzt stimmt's.

Bei diesem Beispiel kann ich Ihnen noch etwas zeigen, diesmal eine Eigenheit - oder besser gesagt eine Vorsorge - des guten alten Editors. Schauen Sie sich genau an, wie der Editor die drei Zahlen 375, 15 und 5 auf den Bildschirm geschrieben hat. Vor der ersten Zahl steht eine Leerstelle, zwischen den Zahlen sind zwei Leerstellen.

Das kommt daher, daß der Editor vor jeder Zahl eine Stelle reserviert, um im Fall einer negativen Zahl Platz für das Minuszeichen zu haben. Und zwischen den Zahlen hält er natürlich einen Abstand frei. Wie bereits gewohnt, wollen wir das am Computer überprüfen.

→ Bitte geben Sie die drei Werte unserer Variablen als negative Zahlen, also mit einem Minuszeichen ein.
Wenn Sie die Zuweisungen noch auf dem Bildschirm stehen haben, geht es schneller mit Hilfe des Editors. Fahren Sie mit dem Cursor auf die jeweils vorderste Ziffer, schaffen Sie mit der INSERT-Taste (INST/DEL + SHIFT) eine Leerstelle und tippen Sie ein Minuszeichen ein, danach wie immer ein RETURN.
Fahren Sie mit dem Cursor auf die Zeile mit dem PRINT-Befehl und drücken Sie die RETURN-Taste.

Ja, der Editor macht es uns wirklich sehr bequem. Und jetzt sehen Sie auch, daß die reservierten Plätze im Ergebnis von den Minuszeichen belegt worden sind. Was bleibt, sind lediglich die Abstände - der guten Lesbarkeit zuliebe.

Das Semikolon ist nicht das einzige Symbol zur Trennung von Variablen hinter einem PRINT-Befehl. Die von Commodore verwendeten Versionen von Basic erlauben auch das Komma - allerdings hat es eine etwas andere Wirkung als das Semikolon.

→ Ersetzen Sie in der Zeile oben, welche den »dreifachen« PRINT-Befehl enthält, die zwei Semikolon durch Komma (direkt durch Überschreiben). RETURN= Drücken nicht vergessen!

Wir sehen jetzt wieder die drei Zahlen, aber viel weiter auseinandergerückt.

Dabei wird der Bildschirm sozusagen in 4 Teile geteilt. Die ausgedruckten Werte beginnen am linken Rand, dann ab dem 10., 20., und 30. Platz.

FAZIT

Hinter einem einzigen PRINT-Befehl können mehrere Werte auf einmal ausgedruckt werden. Sie müssen entweder durch ein Semikolon oder ein Komma getrennt sein.

Das Semikolon bewirkt bei Zahlen einen »normalen« Abstand von 2 Leerstellen.

Das Komma verschiebt die Werte auf das nächste Bildschirm-Viertel.

Buchstaben ausdrucken

Bisher haben wir nur mit Zahlen gearbeitet.

Was aber macht der PRINT-Befehl mit allen anderen Zeichen und Symbolen? Versuchen Sie es ruhig einmal. Fangen Sie mit den Tasten links oben an und arbeiten Sie sich durch.

Schon bei dem PRINT-Befehl der allerersten Taste, nämlich dem Symbol »Pfeil nach links« reagiert der Computer unerwartet.

```
→ PRINT ←  (RETURN drücken)
```

ergibt eine englische Meldung auf dem Bildschirm:

```
SYNTAX ERROR
READY
```

Mit der Meldung »SYNTAX ERROR« sagt uns der Computer, daß wir einen Rechtschreib- oder Satzbaufehler begangen haben. Das Zeichen ← darf zum Beispiel nicht so einfach mit einem PRINT-Befehl verwendet werden.

Alle mathematischen Zeichen, Interpunktionen, die geSHIFTeten Symbole der Zifferntasten und andere reagieren in dieser Form.

Andere wieder, wie zum Beispiel mit SHIFT- oder Commodore-(C=)-Taste umgeschalteten Buchstaben (auf die auf der Vorderseite der Tasten angegebenen Grafiksymbole), ergeben gar keine Reaktion.

Den PRINT-Befehl mit Buchstaben haben wir ja schon mehrfach verwendet. In dem Beispiel ganz oben hat er allerdings nicht den Großbuchstaben A, sondern den Zahlenwert ausgedruckt, der unter dem Kennbuchstaben A im Computer gespeichert worden ist.

Versuchen Sie es mit einem anderen Buchstaben oder Wort, welchem wir noch keinen Wert zugewiesen haben, zum Beispiel:

```
→ PRINT WORT (RETURN drücken)
```

Wir erhalten eine Null.

Das heißt nichts anderes, als daß der PRINT-Befehl schlicht und einfach jeden Buchstaben bzw. jede Buchstabenfolge als ein Stichwort – als eine Variable – interpretiert und deren Wert im Speicher sucht, um sie auszudrucken. Haben wir ihr noch keinen Wert zugewiesen, wie in unserem letzten Beispiel dem Wort »WORT«, dann druckt er eine Null aus.

Wenn der für den Bildschirm zuständige Editor (ein fest eingebautes Computerprogramm) bereitwillig alle Buchstaben druckt, die wir direkt mit der Tastatur eingeben, dann muß es der Computer mit dem PRINT-Befehl auch können.

Die Lösung für unser Problem könnte man als zungenbrechendes Kochrezept so hinschreiben:

Um den Computer mitzuteilen, daß das Wort »WORT« ein Wort sein soll, muß man »WORT« schreiben. Das Geheimnis liegt im Gebrauch der Gänsefüße!

```
→ PRINT "WORT" (RETURN drücken)
```

Zwischen Gänsefüßen können Sie auch alle grafischen Symbole der Buchstabentasten, die auf der Vorderseite der Tasten zu sehen sind, per PRINT auf den Bildschirm bringen.

Für die linken Symbole müssen wir mit der Taste gleichzeitig die Commodore-Taste (unten ganz links), für die rechten Symbole die SHIFT-Taste drücken.

Die Schwierigkeit für mich liegt nun darin, daß ich Ihnen im Text des Kurses diese speziellen Symbole nicht ohne weiteres angeben kann. Diese Commodore-spezifischen Zeichen und Symbole hat der Setzer des Verlages verständlicherweise nicht zur Verfügung.

Daher werde ich an ihrer Stelle folgende Symbole anwenden:

- für die SHIFT-Taste............(SHIFT)
- für die Commodore-Taste...........(C=)
- (SHIFT W) bezeichnet das Symbol vorn rechts auf der W-Taste
- (C= Z) bezeichnet das Symbol vorn links auf der Z-Taste

Ich zeige es am besten an einem Beispiel:

```
→ PRINT "(SHIFT U)(SHIFT I)" (RETURN drücken)
```

dann druckt der Computer aus den beiden Viertelkreisen der U- und I-Taste einen Halbkreis aus.

Noch ein Beispiel:

```
→ PRINT "(C= Q)(C= W)" (RETURN drücken)
```

ergibt ein großes H.

Der Gänsefuß-Modus

Die Schreibweise zwischen Gänsefüßen erlaubt uns, nicht nur Buchstaben und deren umgeschaltete grafische Zeichen, sondern auch Ziffern und alle Symbole auszuPRINTen.

Verzeihen Sie mir bitte dieses saloppe Computerdeutsch. Eigentlich ist es ja nicht sehr schön, was wir da mit der deutschen Sprache machen und ich müßte mich eigentlich schämen, daß ich derartige Wortschöpfungen verwende. Aber abgesehen davon, daß Computerdeutsch gewisse Vorgänge sehr kurz und präzise ausdrückt, verteidige ich mich gern damit, daß andere Gruppen unserer Gesellschaft ja auch ihre eigenen Ausdrücke haben, wie die Segler, Musiker und Jäger. Bei den letzteren meine ich natürlich nicht das Jägerlatein.

Wo waren wir stehen geblieben? Ach ja, beim ausPRINTen von Ziffern und Symbolen.

```
→ PRINT "()" (RETURN drücken)
```

Diese Anweisung druckt die beiden Klammern ohne SYNTAX ERROR aus. Alle Pfeile, Sterne, Interpunktionen und arithmetischen Symbole sind willig ausdruckbar, wenn sie in Gänsefüße eingepackt sind.

Apropos Interpunktionen! Wir haben gelernt, daß mit dem Semikolon Stichwörter (Variable) voneinander getrennt werden können, selbst wenn sie hintereinander bei derselben PRINT-Anweisung stehen.

Zwischen Gänsefüßen verlieren sie diese Wirkung; sie sind dann nichts anderes als grafische Zeichen. Geben Sie ein:

```
→ A = 3  (RETURN)
→ B = 12 (RETURN)
→ C = 17 (RETURN)
→ PRINT A;B;C;"A;B;C" (RETURN)
```

Zuerst wird der Wert der drei Variablen A, B und C nebeneinander ausgedruckt, gefolgt von den 5 Symbolen zwischen den Gänsefüßen.

Mit dem Komma und seiner verschiebenden Wirkung ist es ebenso.

Und Ziffern? Ziffern gehen auch, aber sie gingen ja vorher schon, ohne Gänsefüße.

Da ist aber ein ganz kleiner Unterschied. Passen Sie auf:

```
→ PRINT 123   (RETURN)
→ PRINT "123" (RETURN)
```

- Der erste PRINT-Befehl versteht die Ziffern 123 als Zahl und reserviert, wie früher schon besprochen, eine Stelle vorher für das Vorzeichen.
- Der PRINT-Befehl mit Gänsefüßen versteht die Ziffern 123 als grafische Symbole, die natürlich kein Vorzeichen brauchen und daher auch keine Platzreservierung bekommen.

Dieser feine Unterschied ist recht wichtig, wenn Zahlenta-

bellen oder Zahlengrafiken erzeugt werden sollen. Wir stoßen später sicher noch darauf.

Bevor ich ein weiteres Fazit ziehe, möchte ich Ihnen eine – vorläufig – letzte Anwendung des PRINT-Befehls zeigen.

Wir haben gesehen (und angewendet), daß eine auf den Bildschirm geschriebene Zeile erst dann zu einer Anweisung für den Computer wird, wenn sie mit dem Drücken der RETURN-Taste »abgeschlossen« wird. Wir haben daher immer nur einen einzigen Befehl in eine Zeile geschrieben.

Die Frage stellt sich natürlich:

Können wir mehrere Befehle in eine einzige Zeile schreiben??

Wie immer – probieren geht über studieren! Geben Sie ein:

→ PRINT "1.BEFEHL"   PRINT "2.BEFEHL"
→ drücken Sie die RETURN-Taste

Nun, SYNTAX ERROR sagt uns, daß diese Schreibweise nicht erlaubt ist.

Wie bei mehreren Variablen brauchen wir bei mehreren Befehlen auch ein Symbol zur Trennung. Bei Befehlen ist dies der Doppelpunkt. Fügen Sie bitte zwischen die beiden PRINT-Befehle oben einen Doppelpunkt ein und drücken Sie nochmal auf Return.

Es muß also dastehen:

→ PRINT "1.BEFEHL" : PRINT "2.BEFEHL"

Jetzt stehen die beiden »Befehle« ausgedruckt da – untereinander!

Durch den Doppelpunkt werden die zwei PRINT-Befehle so behandelt, als stünden sie in zwei getrennten Zeilen.

Und damit wird auch unsere anfängliche Rechnerei wesentlich einfacher – ich hoffe, Sie erinnern sich noch. Sonst schauen Sie bitte nochmal am Anfang nach.

Für die Berechnung der Summe 3 + 2 brauchen wir nur eine Zeile:

→ SUMME = 3 + 2 : PRINT SUMME (RETURN)

Damit soll's erstmal genug sein. Ich glaube, Sie haben jetzt eine ganze Reihe von Regeln gelernt, die für die ersten Schritte der Programmierung eines Commodore-Computers sehr wichtig sind.

FAZIT

1. Buchstaben werden vom PRINT-Befehl als Variable eingestuft, und entsprechend wird ihr Wert ausgedruckt.
2. Ziffern und Zahlen werden vom PRINT-Befehl als arithmetische Ausdrücke gewertet und erhalten dementsprechend ein Vorzeichen. Das positive Vorzeichen wird nicht ausgedruckt, eine Leerstelle ist dafür vorhanden.
3. Alle Symbole der Tastatur, welche zwischen Gänsefüßen stehen, werden vom PRINT-Befehl als graphische Zeichen erkannt und formgetreu ausgedruckt.
4. Um mehrere Befehle in eine einzige Zeile schreiben zu können, müssen sie durch einen Doppelpunkt voneinander getrennt sein.
5. Schreibfehler oder inkorrekte Anweisungen quittiert der Computer mit der Fehlermeldung SYNTAX ERROR.

Was ist ein Programm?

Bislang haben wir den Computer so benützt, als wäre er ein Taschenrechner.

Wir haben alle Angaben, Befehle und so weiter direkt eingetippt; und der Computer hat sie nach dem Drücken der RETURN-Taste direkt ausgeführt.

Ich habe mit Absicht das Wort direkt gleich zweimal verwendet. Diese Art des Betriebs wird nämlich DIREKT-MODUS genannt.

Im Gegensatz dazu steht der sogenannte PROGRAMM-MODUS.

Jetzt also taucht das Schlagwort Programm zum ersten Mal auf. Haben Sie schon sehnlich darauf gewartet? Es wäre kein Wunder, wollen wir doch Programmieren lernen.

In meinem alten Konversationslexikon steht unter Programm: Festfolge. Sie wissen, was damit gemeint ist, nämlich eine detaillierte Angabe der einzelnen Darbietungen und Aktivitäten einer Veranstaltung in ihrer genauen zeitlichen Reihenfolge.

Diese Definition läßt sich gut auf einen Computer übertragen.

Wenn ein Computer viele Befehle in einer bestimmten Reihenfolge nacheinander ausführen soll, müssen sie ihm als »Festfolge« vorgegeben sein. Diese Festfolge muß er auswendig kennen – was für ihn kein Problem bedeutet, besitzt er doch ein gutes Gedächtnis.

Dieses »Gedächtnis« haben wir ja schon kennengelernt. Die im Direkt-Modus eingegebenen Werte von Stichwörtern (Variablen) hat er in seinem Variablen-Speicher festgehalten und wir konnten sie jederzeit abfragen.

Ein Rechner hat aber auch einen Programm-Speicher, in welchem er ein aus mehreren Befehlen bestehendes Programm festhält.

Alles, was wir tun müssen, ist ihm die Befehle und ihre Reihenfolge einzugeben. Wie tun wir das?

Die Angabe der Reihenfolge ist sehr simpel. Wir geben einfach jedem Befehl eine Nummer.

Die Eingabe der Befehle geht genauso wie vorher – mit der RETURN-Taste. Nur – vorher hat der Computer den Befehl sofort ausgeführt. Wenn aber eine Nummer davorsteht, dann führt er den Befehl nicht aus, sondern steckt ihn in seinen Programmspeicher.

Das machen wir jetzt gleich einmal. Geben Sie bitte unsere allerersten Rechenanweisungen mit Befehlsnummern ein:

→ 1 SUMME = 3 + 2 (RETURN-Taste drücken)
→ 2 PRINT SUMME   (RETURN)

Siehe da, er hat nichts ausgeführt und ich behaupte, dieses kleine Programm steht jetzt im Programmspeicher.

Ich kann das auch beweisen. Es gibt einen Befehl in Basic, der alles, was im Programmspeicher steht, in der Reihenfolge der Befehlsnummern auf dem Bildschirm ausdruckt. Er heißt LIST

was auf deutsch soviel wie »auflisten« bedeutet.

→ Löschen Sie den Bildschirm mit der CLR-Taste
→ Tippen Sie direkt das Wort LIST ein und drücken Sie die RETURN-Taste.

Quod erat demonstrandum (auf deutsch: »Was zu beweisen war«).

Die beiden Anweisungen samt Nummer erscheinen auf dem Bildschirm.

Basic-Befehl Nr. 3 LIST

- druckt den Inhalt des Programmspeichers auf dem Bildschirm aus.
- Durch Angabe der Nummer oder eines Nummernbereiches können einzelne Befehle oder bestimmte Programmteile separat ausgedruckt werden.
- Durch den Befehl LIST wird das Programm nicht gelöscht.

Den ersten Satz dieser Befehlsbeschreibung haben wir ja schon praktiziert. Den zweiten Satz möchte ich vorläufig erst mal so stehen lassen. Zu seiner Erprobung brauchen wir längere Programme und die haben wir vorerst noch nicht.

So, unser Programm, bestehend aus zwei Befehlen, hat der Computer gespeichert. Jetzt soll er es ausführen!

Unsere alte Methode, den Computer hinter dem Ofen hervorzuscheuchen, war der Tritt mit der RETURN-Taste. Diese bringt uns hier aber überhaupt nichts. Was wir brauchen ist ein Befehl, der den Computer veranlaßt, das Programm auszuführen. Ein derartiger Befehl wird uns von Basic geboten.

Er heißt RUN

was mit »loslaufen« übersetzt werden könnte.

Basic-Befehl Nr. 4 RUN
- startet ein im Programmspeicher des Computers befindliches Programm.
- Mit RUN wird ein Programm nicht gelöscht. Der RUN-Befehl kann beliebig oft wiederholt werden.

⟶ Tippen Sie direkt das Wort RUN ein und drücken Sie die RETURN-Taste.

Und wie der Blitz, so schnell können Sie gar nicht schauen, steht das Resultat auf dem Bildschirm.

Na ja, sagen vielleicht einige Skeptiker unter Ihnen, im Direkt-Modus vorhin ging das auch nicht viel langsamer, besonders die Einzeiler-Version mit dem Doppelpunkt.

Richtig, aber wir haben ja nur zwei Befehle verwendet. Nehmen Sie mal 20 Befehle, da sieht die Sache schon anders aus. Abgesehen von der Geschwindigkeit gehen die überhaupt nicht in eine Zeile - und was dann?

FAZIT
1. Der Computer hat einen Speicher mit mehreren getrennten Bereichen. Alle Variablen und ihre jeweiligen Werte stehen im Variablenspeicher. Programmzeilen stehen im Programmspeicher.
2. Immer wenn die RETURN-Taste gedrückt wird, überprüft der Computer die Zeile, in der sich gerade der Cursor befindet. Beginnt die Zeile mit einer Nummer, wird sie als Programmzeile erkannt und im Programmspeicher abgelegt.
   Hat sie keine Nummer am Anfang, dann führt sie der Computer im Direkt-Modus sofort aus.
3. Mit dem Basic-Befehl LIST wurden alle im Programmspeicher stehenden Zeilen in Reihenfolge der aufsteigenden Zeilennummern auf dem Bildschirm ausgedruckt.
4. Ein Programm wird mit dem Basic-Befehl RUN gestartet.

Nun sollten wir uns schleunigst an ein längeres Programm wagen. Natürlich verwenden wir nur Dinge, die wir bisher schon verwendet beziehungsweise besprochen haben.

Das erste richtige Programm

Zuerst brauchen wir einen Plan, was programmiert, oder besser gesagt, was vom Computer gemacht werden soll.

Ich schlage dazu folgende Aufgabenstellung vor:

a) Unter dem Stichwort A sollen zwei Zahlen multipliziert werden
b) Unter dem Stichwort B sollen zwei Zahlen dividiert werden
c) Die Variable (das Stichwort) C sei die Differenz der Werte von A und von B
d) Der Wert von A wird nicht als Zahl, sondern als Gleichung ausgedruckt.
f) Dann folgt eine Textangabe, wie der Wert von C berechnet wird. Dabei sollen im Text die jeweiligen Werte von A und B automatisch eingesetzt werden.
g) Als letztes wird auch der Wert von C als Gleichung ausgedruckt.

Das klingt schlimmer, als es ist. Wir müssen es einfach Schritt für Schritt angehen. Für Punkt a nehmen wir die Zahlen 24 und 3

⟶ `1 A = 24 * 3 (RETURN)`

In dieser Zeile mit der Nummer 1 weisen wir den Variablen A das Resultat der Multiplikation von 24 mal 3 zu. Wir hätten das auch mit dem Basic-Befehl LET machen können, aber wie gesagt, man kann ihn weglassen.

Punkt b verlangt etwas Ähnliches, wir nehmen die Zahlen 16 und 8.

⟶ `2 B = 16/8 (RETURN)`

Auch Punkt c macht keine Ausnahme, nur verwenden wir jetzt statt absoluter Zahlen die Variablen A und B. Der Computer muß selbst im Lauf des Programms die Werte für A und B im Variablenspeicher heraussuchen und einsetzen.

⟶ `3 C = A - B`

Bis hierhin war alles so wie gehabt. Im Punkt d soll der Wert von A, also das Resultat aus Zeile 1 als »Gleichung« ausgedruckt werden.

Was ich damit meine, ist schnell gesagt. Der Befehl PRINT A würde nur den Zahlenwert von A auf den Bildschirm bringen. Ich möchte aber, daß das Programm hinschreibt:
A = 72

Wie erreichen wir das ?

Nun, die Zahl 72 ist der Wert von A, mit PRINT A zu erzielen. Davor steht ein Text! Damit der PRINT-Befehl A = hinschreibt, müssen wir den Gänsefuß-Modus anwenden. Das sieht dann so aus:

⟶ `4 PRINT "A =" A (RETURN)`

Was zwischen den Gänsefüßen steht, druckt der Computer so aus, wie es ist. Das zweite A steht nackt da, also ist es eine Variable, deren Wert der Computer direkt hinter den Text »A =« setzt - mit Vorzeichenstelle natürlich.

Wenn Sie es nicht glauben oder es sich nicht vorstellen können, lassen Sie doch diesen ersten Teil des Programms gleich einmal laufen. Zur besseren Übersicht löschen Sie zuerst den Bildschirm mit der .....-Taste (nicht raten, wissen!).

Keine Angst, es wird nur der Bildschirm gelöscht, aber nicht das Programm im Programmspeicher. Um das nachzuprüfen, LISTen Sie es aus. Wenn das Programm schön dasteht, lassen Sie es mit RUN und RETURN-Taste laufen.

Auf Ihrem Bildschirm steht jetzt unter der Programm-Liste:
A=72

Das ist also das Resultat der Zeile 4. Genauso machen wir es mit dem Punkt e.

⟶ `5 PRINT "B = " B (RETURN)`

Punkt f ist reiner Text, den ich in zwei PRINT-Befehle aufteile, damit der Text in zwei getrennten Zeilen steht.

⟶ `6 PRINT "C ist A MINUS B,"`
⟶ `7 PRINT "DAS HEISST," A "WIRD UM" B "VERRINGERT."`

Zeile 6 besteht aus reinem Text - alles steht zwischen Gänsefüßen. Zeile 7 ist wieder etwas komplizierter, aber sie enthält im Grunde genommen nichts anderes als die Zeilen 4 und 5 vorher. Man muß nur die Gänsefüße richtig abzählen. Zwischen den ersten beiden wird Text ausgedruckt, inklusive dem Komma. Dann folgt der Wert der Variablen A - ohne Gänsefüße natürlich. Zwischen den nächsten beiden Textteilen steht wieder einsam die Variable B, deren Wert der Computer suchen und hier einsetzen muß.

Auch jetzt ist es Ihnen schon erlaubt, einen vorläufigen Probelauf zu machen.

Schließlich führen wir noch Punkt g aus, wofür wir wieder die Methode der Zeilen 4 und 5 anwenden:

⟶ `8 PRINT "C =" C`

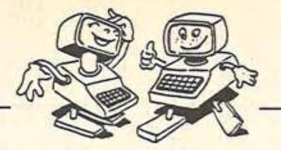
Ich habe übrigens ganz klammheimlich den ursprünglich verwendeten Begriff der »Befehlsnummer« in »Zeilennummer« umgetauft. Haben Sie es gemerkt?

Zeilennummer ist tatsächlich der gängige Ausdruck, wohl daher, weil in einer Zeile mehrere Befehle stehen können, durch Doppelpunkte getrennt. Ich werde also im folgenden immer Zeilennummer sagen.

Es empfiehlt sich immer, ein Programm nochmal als Ganzes auszuLISTen. Also: Bildschirm löschen und LIST eingeben.

Wir sehen dann auf dem Bildschirm:

```
1 A = 24 * 3
2 B = 16/8
3 C = A - B
4 PRINT "A =" A
5 PRINT "B =" B
6 PRINT "C IST A MINUS B,"
7 PRINT "DAS HEISST,"A "WIRD UM" B
"VERRINGERT"
8 PRINT "C =" C
```



Alle diejenigen, welche den PRINT-Befehl mit dem Fragezeichen abgekürzt haben, werden jetzt staunen. Trotz der Abkürzerei steht im Programm-Ausdruck – auch Listing genannt – kein Fragezeichen, sondern das volle Wort »PRINT«. Das ist wieder die Tat des guten alten Editors, der flugs alles in seine rechte Bahn umlenkt und umkodiert.

Wenn Sie beim Tippen keine Fehler gemacht haben, dann dürfen Sie das Programm laufen lassen.

Wenn Sie einen Fehler entdeckt haben, dann nutzen Sie einfach den Editor aus und korrigieren Sie den Fehler direkt in der Zeile, wo er auftritt. Nach der Reparatur nicht die Zeile verlassen, sondern zuerst RETURN drücken, damit die neue, korrigierte Zeile an die Stelle der alten falschen Zeile, die ja dieselbe Nummer hat, gespeichert wird.

So, jetzt hindert uns aber niemand mehr am RUN (und natürlich RETURN-Taste).

Auf Ihrem Bildschirm muß jetzt folgendes Resultat stehen:

```
A= 72
B= 2
C ist A MINUS B,
DAS HEISST, 72 WIRD UM 2 VERRINGERT
C= 70

READY
```

Ja, was wollen Sie machen, wenn das Programm noch immer nicht das tut, was es soll?

Als allererstes nicht verzweifeln! Tippfehler treten immer wieder auf, die der Computer in seiner Perfektion eben sofort merkt. Da er aber keine eigene Intelligenz hat, tut er wirklich nur genau das, was wir ihm anschaffen – mit allen Fehlern.

Eine Fehlersuche kann oft sehr mühsam sein, besonders bei langen und komplexen Programmen.

Nun, in unserem Fall dürfte es nicht allzu schwer sein, Zeile für Zeile alle Buchstaben und Symbole durchzugehen, bis Sie den Fehler gefunden haben.

Später werden wir noch Methoden der Fehlersuche kennenlernen. Heute möchte ich Sie noch nicht damit belasten.

Ich möchte vielmehr, daß Sie noch ein bißchen mit dem Programm spielen. Verändern Sie die Zahlenwerte in den Zeilen 1 und 2 durch direktes Überschreiben – aber immer mit RETURN abschließen, sonst bleibt der alte Wert erhalten.

Sie brauchen dann nicht erneut RUN eintippen, es steht ja noch da. Fahren Sie mit dem Cursor drauf und drücken Sie die RETURN-Taste. Sofort wird das Programm wieder ausgeführt und wie mit Geisterhand erscheinen neue Zahlen an den Stellen der Variablen.

Aber Achtung!! Die Werte auf dem Bildschirm werden nur überschrieben. Wenn der neue Wert kürzer ist als der alte, dann bleibt der »längere« Rest des alten Wertes stehen; das kann zur Verwirrung führen.

Das passiert dadurch, daß dem Computer ja keine Anweisung gegeben worden ist, vor der Berechnung den Bildschirm zu löschen. Diese Anweisung kommt erst später in unser Lehrprogramm.

Ein anderes »Experiment« besteht darin, die Zeilennummer einer Befehlszeile zu ändern.

⟶ Fahren Sie mit dem Cursor auf die 8 dieser Zeile
Überschreiben Sie die 8 mit der Ziffer 9 und drücken Sie Return

Wir haben jetzt eine neue Programmzeile geschaffen, mit identischem Inhalt wie die alte Zeile 8.

⟶ Geben Sie LIST und RETURN ein.

Siehe da, wir haben eine neue Zeile 9, aber die alte Zeile 8 ist auch noch da, natürlich, denn wir haben sie ja durch das Überschreiben nicht gelöscht.

Das Löschen einer Zeile geht ganz einfach:

⟶ Tippen Sie die Nummer der Zeile, in unserem Falle eine 8 und drücken Sie RETURN

Wir haben also eine Zeile 8 ohne Text beziehungsweise ohne Befehle eingegeben. Da dem Rechner sein Speicherplatz sehr kostbar ist, ignoriert er eine solche nichtssagende Zeile – sie ist weg. Überprüfen Sie es mit LIST.

Jetzt haben wir ein Programm, in dem in der Reihenfolge der Zeilen eine Nummer fehlt. Was macht das? Mit RUN können Sie sich überzeugen, daß das dem Computer völlig egal ist. Er braucht nämlich nur Zeilennummern in aufsteigender Folge, Lücken überspringt er einfach.

Das bedeutet aber, daß wir unser Programm oben auch mit den Zeilennummern 10..20..30.. und so weiter hätten schreiben können. In der Tat möchte ich Ihnen diese Vorgehensweise empfehlen, da sie ein späteres Einfügen von weiteren Programmzeilen erlaubt. Wir werden das noch üben.

FAZIT

1. Innerhalb eines PRINT-Befehls können Variable und Zeichen beliebig miteinander gemischt werden. Wichtig ist nur, daß Zeichen immer innerhalb von Gänsefüßen stehen müssen.
2. Ein bereits im Progammspeicher stehendes Programm kann auf dem Bildschirm jederzeit abgeändert werden. Es gelten bei dem »Überschreiben« alle Regeln und Möglichkeiten des Editors. Eine Änderung muß mit Return abgeschlossen werden.
3. In dem Programmspeicher wird die jeweils letzte mit RETURN eingegebene Version einer Zeile gespeichert.
4. Zeilennummern lassen sich auf dieselbe Art und Weise verändern. Eine Zeile wird durch Eingabe lediglich der Zeilennummer, also ohne Text, gelöscht.
5. Zeilennummern brauchen nicht direkt aufeinander zu folgen. Lücken in der Zahlenfolge werden übersprungen.
6. Es empfiehlt sich, ein Programm mit Zeilennummern zu versehen, die zum Beispiel jeweils um 10 voneinander verschieden sind. Das erleichtert das spätere Einfügen von zusätzlichen Programmzeilen.

Eingabe – oder wie sage ich es meinem Computer

Wir haben ein erstes kleines Programm geschrieben. Aber sein Wortschatz ist noch sehr begrenzt – LET und PRINT. Die beiden anderen Befehle LIST und RUN werden vorläufig nur im Direkt-Modus verwendet. Bevor wir unseren Basic-Wortschatz erweitern, empfehle ich Ihnen, das alte Programm aus dem Speicher des Computers zu entfernen und ihn frei zu machen für ein neues Programm.

Verwechseln Sie das bitte nicht mit dem Löschen des Bildschirms. Zur Klärung:

- Die CLR/HOME-Taste geSHIFTet löscht den Bildschirm; ein Programm im Speicher bleibt davon unberührt.
- Durch Aus- und Wiedereinschalten des Computers löschen Sie sowohl den Bildschirm als auch den Speicher des Computers, in welchem das Programm gespeichert ist. Der Computer meldet sich dann mit seinen Einschalt-Überschriften.
- Es gibt auch einen Basic-Befehl, der ein Programm aus dem Speicher hinauswirft, ohne den Bildschirm zu löschen. Er heißt

NEW

Probieren Sie es bitte aus.

⟶ Geben Sie eine beliebige Programmzeile (mit Nummer) ein, zum Beispiel:
`20 PRINT"ABCDE"`

⟶ löschen Sie mit der DEL-Taste den Bildschirm
geben Sie direkt LIST ein

Jetzt erscheint die obige Zeile 20 wieder.

⟶ geben Sie direkt NEW ein und nach der RETURN-Taste wieder den Befehl LIST



Die Programmzeile 20 ist verschwunden

Basic-Befehl Nr.5 NEW
- macht den Programmspeicher frei für die Eingabe eines neuen Programms
- ein »altes« im Speicher sitzendes Programm wird dadurch zwar nicht gelöscht, aber es kann nach dem NEW-Befehl nur noch unter ganz bestimmten Umständen und nur mit besonderen Tricks wieder verfügbar gemacht werden

Seien Sie also vorsichtig mit diesem Befehl, immer erst überlegen, bevor Sie NEW mit der RETURN-Taste abschließen, hat er doch eine so gut wie endgültige Wirkung. Salopp ausgedrückt »löscht« er den Programmspeicher.

Auch dieser Befehl wird fast ausschließlich nur im Direkt-Modus eingesetzt.

So, jetzt aber möchte ich Sie mit der Möglichkeit vertraut machen, während des Ablaufs eines Programms – also nicht schon beim Eintippen desselben – dem Computer Anweisungen in Form von Zahlen und Wörtern zu geben.

Bislang haben wir, wie gesagt, alle Zahlen und Wörter, die der Computer auf den Bildschirm bringen sollte, immer vorher, das heißt, beim Schreiben der Programmzeilen eingegeben, als Wert-Zuweisung für eine Variable (das gute alte Stichwort), mit und ohne LET-Befehl.

Dasselbe während des Ablaufs eines Programms ermöglicht der BASIC-Befehl

INPUT (gefolgt von einer Variablen)

Auf deutsch würde dieser Befehl »Eingabe« heißen.

Im Direkt-Modus können wir ihn leider nicht einsetzen. Versuchen Sie es ruhig:

```
→ INPUT A (RETURN nicht vergessen!)
```

Der Computer bestraft uns mit der Fehlermeldung »ILLEGAL DIRECT«. Aber im Programm-Modus, das heißt mit einer Zeilennummer versehen, geht es. Auch das wollen wir ausprobieren:

```
→ 10 INPUT A     (RETURN)
```

Jetzt nur noch RUN eingeben, und siehe da, der Computer druckt ein Fragezeichen und wartet mit blinkendem Cursor.

Daß er wirklich wartet, merken Sie daran, daß er beharrlich immer wieder die Fehlermeldung »REDO FROM START« – was einfach »NOCH EINMAL« heißt – und danach das auffordernde Fragezeichen ausdruckt, wenn Sie einen Buchstaben oder ein Zeichen eingeben.

Er akzeptiert nur Zahlen, und außerdem noch die Leertaste, den Punkt, das Plus- und das Minuszeichen. Die RETURN-Taste bricht den Befehl ab, und das Programm fährt mit der nächsten Zeile fort.

Warum akzeptiert »INPUT A« nur Zahlen? Was passiert bei INPUT eigentlich?

Die Zahl, die wir per Tastatur eingeben, wird der Variablen A, die hinter dem Befehlswort INPUT steht, zugeordnet – genauso wie mit dem LET-Befehl.

Mit der folgenden Programmzeile, die Sie zusätzlich zur Zeile 10 eingeben, wird diese Zuordnung bewiesen:

```
→ 10 INPUT A
→ 20 PRINT A
```

Die beiden Zeilen 10 und 20 zusammen ergeben ein Mini-Programm, welches nach RUN auf eine Eingabe wartet und diese – sofern es eine Zahl ist – auf dem Bildschirm ausdruckt.

Wir können also in der Tat mit der Tastatur innerhalb eines Programms Zahlenwerte in dasselbe eingeben. Aber warum nur Zahlen?

Die Antwort ist einfach: Einer Variablen, die den Namen »A«, oder »ZAHL« oder »SUMME« hat, dürfen nur Zahlen zugeordnet werden.

Das heißt aber nicht, daß wir auf schriftliche Anweisungen, die aus Buchstaben, Zeichen oder ganzen Wörtern bestehen, verzichten müssen. Der INPUT-Befehl akzeptiert sie auch, aber nur, wenn wir eine andere Art von Variable benützen.

Das ist ein neues Gebiet für uns. Erinnern Sie sich, beim LET-Befehl habe ich gesagt, daß ich die Variablen später noch genauer behandeln würde. Jetzt ist es soweit.

Es gibt mehrere Arten von Variablen

Ich habe Ihnen erklärt, daß wir mit Variable die Stichwörter bezeichnen, unter denen der Computer sich Zahlenwerte merkt und unter denen sie auch wieder aus dem Speicher hervorgeholt werden können.

Des öfteren wird das alles mit Schachteln verglichen, auf die der Name einer Variablen geschrieben wird. Ein Wert, welcher der Variablen zugeordnet wird, kommt in die Schachtel mit dem richtigen Namen hinein. Eine Kopie des Wertes kann jederzeit aus der Schachtel herausgeholt werden, wenn man ihren Namen kennt. Der Wert bleibt solange in der Schachtel, bis er durch einen neuen ersetzt wird.

Es gibt mehrere Arten von Variablen, von denen wir zwei besprechen wollen.

1. Numerische Variable

Diesen Typ haben wir bislang verwendet. Er enthält nur Zahlen: ganze Zahlen (325), Dezimalbrüche (0.325) und negative Zahlen (-325, -0.325).

Bei der Wahl des Namens der Variablen, der vorn auf die Schachtel geschrieben wird, sind wir ziemlich frei. Er muß nur immer mit einem Buchstaben anfangen. Die nachfolgenden Zeichen können Buchstaben, Ziffern oder grafische Zeichen sein.

Wir haben bisher X, SUMME, PRODUKT etc. verwendet. Wir dürfen aber auch B23X oder F5 nehmen.

Der Länge des Variablen-Namens ist theoretisch nur durch die Anzahl der verfügbaren Zeichen in der Programmzeile eine Grenze gesetzt. Praktisch ist das aber ohne Bedeutung, da der Computer nur die ersten beiden Zeichen als Namen verwendet. So vielsagend zum Beispiel in einem Programm die Variablen-Namen »PRODUKT« und »PROFIT« sein könnten, der Computer erkennt nur »PR« und nimmt an, es handle sich um dieselbe Variable. Also Vorsicht !!!

2. String-Variable

So nennen wir den zweiten Typ. Hier geht es nun um Schachteln, in die wir Buchstaben, Zeichen, Wörter, ja sogar ganze Sätze bis zu einer maximalen Länge von 255 Zeichen hineingeben dürfen.

Diese Aneinanderreihung von Zeichen bezeichnen wir mit »Zeichenketten« oder mit dem englischen Wort »String«, das sich wegen seiner Kürze allgemein durchgesetzt hat. Daher auch der Name »String-Variable«.

Der Name einer String-Variablen, der auf der Schachtel steht, ist genauso aufgebaut wie der einer numerischen Variablen, nur muß am Ende immer das Dollar-Zeichen »$« – die geSHIFTete 4-Taste – stehen.

Der Vergleich der beiden Variablentypen sieht so aus:

```
→ 10 A = 25
  20 A$ = "ZEICHENKETTE"
  30 PRINT A
  40 PRINT A$
```

Sie sehen, selbst bei gleichem Namen, nämlich A, unterscheidet der Computer exakt zwischen der numerischen Variable A und der Stringvariable A$.

Übrigens, beim PRINT-Befehl haben wir diesen Unterschied auch schon gemacht. Alles, was wir nämlich damals zwischen Gänsefüßen geschrieben haben, war nichts anderes als Zeichenketten oder Strings.

Strings und Stringvariable werden uns fortan laufend begegnen. Deswegen schlage ich zur Übung noch ein paar Beispiele vor.

Löschen Sie bitte das obige Programm mit NEW und geben Sie die folgenden Programmzeilen ein:

```
→ NEW
  10 T$="HOLZ"
  20 U$="FEUER"
  30 W$="7"
  40 X$="5"
  50 PRINT U$:PRINT T$
  60 PRINT U$;:PRINT T$
  70 PRINT T$;:PRINT U$
  80 PRINT W$;:PRINT X$
  90 PRINT W$ X$
```

Diese Programmzeilen ergeben auf dem Bildschirm folgendes Bild:

FEUER
HOLZ
HOLZFEUER
FEUERHOLZ
75
75

In den Zeilen 10 bis 40 werden 4 String-Variable definiert und ihnen werden zwei Wörter und zwei Ziffern zugewiesen. Die Ziffern sind nicht Zahlenwerte, sondern auch Strings, da sie zwischen Gänsefüßen stehen.

Zeile 50 enthält zwei – durch den Doppelpunkt getrennte – PRINT-Befehle, die untereinander die Strings der beiden Variablen U$ und T$ ausdrucken.

Zeile 60 ist fast identisch mit Zeile 50, und doch ist ihr Resultat grundlegend verschieden, da beide Strings nebeneinander in einem Wort geschrieben werden. Der winzige Unterschied ist das Semikolon am Ende des ersten PRINT-Befehls. Wir haben das Semikolon schon kennengelernt, als Trennzeichen bei der Aneinanderreihung von mehreren numerischen Variablen innerhalb eines einzigen PRINT-Befehls.

Hier bei den Strings hat es eine entgegengesetzte Wirkung. Es klebt nämlich die Strings zweier Variablen, die in getrennten PRINT-Befehlen stehen, aneinander.

Ich weiß, das sieht verwirrend aus. Aber glauben Sie mir, diese »Rechtschreibregeln« werden Ihnen sehr rasch in Fleisch und Blut übergehen.

Zeile 70 ist ein weiteres Beispiel für den Alleskleber Semikolon, nur diesmal in umgekehrter Reihenfolge der Strings.

Zeile 80 zeigt Ihnen, daß die Ziffern 7 und 5 in der Tat nicht als Zahlenwerte, sondern als Strings behandelt werden. Numerische Variable werden, wie Sie sich erinnern, mit Leerzeichen für das Vorzeichen und für den Abstand ausgedruckt, Strings dagegen nicht. Und da ist es egal, ob der String zufällig eine Ziffer ist.

Zeile 90 schließlich zeigt, daß wie bei den numerischen Variablen auch mehrere String-Variable hinter einen einzigen PRINT-Befehl geschrieben werden können. Da das $-Zeichen das Ende der String-Variablen eindeutig festlegt, kann bei Strings das Semikolon weggelassen werden.

Die nächsten drei Zeilen erweitern das Programm:

```
→ 100 Y$=T$+U$
  110 Z$=U$+T$
  120 PRINT Y$:PRINT Z$
```

Die beiden String-Variablen T$ und U$ werden zu einem neuen String Y$ und in umgekehrter Reihenfolge zu Z$ zusammengebaut (Zeile 100 und 110). Zeile 120 druckt uns das Resultat aus.

Zum Schluß will ich noch meine Behauptung von oben, daß nämlich nur die beiden ersten Zeichen einer Variablen erkannt werden, beweisen.

```
→ 130 PRODUKT=25:PRINT PR
  140 PROFIT= 3:PRINT PR
  150 PRINT PRODUKT
```

Zeile 130 weist der numerischen Variablen »PRODUKT« den Zahlenwert 25 zu und druckt ihn aus, wobei die ersten beiden Buchstaben der Variablen genügen, um sie im Speicher zu finden.

Zeile 140 tut dasselbe für die Variable »PROFIT« mit dem Zahlenwert 3. Und siehe da, die Variable PR, die vorher noch 25 war, wird jetzt mit dem Wert 3 ausgedruckt. Daß wir tatsächlich in Zeile 140 der Variablen PRODUKT einen neuen Wert zugewiesen haben, obwohl wir den Variablen-Namen PROFIT gewählt haben, beweist uns Zeile 150, die unter PRODUKT dennoch die 3 ausdruckt.

Übrigens, wenn Sie nur der Ablauf der letzten drei Zeilen des Programms interessiert, können Sie eine Eigenschaft des Befehls RUN ausnützen:

Wenn Sie
RUN 130
eingeben, läuft das Programm ab Zeile 130 los und läßt alle vorhergehenden Zeilen unbeachtet.

Wenn Sie aber nach dem RUN eine Zeilennummer angeben, die im Programm nicht vorkommt, läuft das Programm nicht los, sondern es meldet sich der Computer mit der Fehlermeldung »UNDEF'D STATEMENT« (undefined statement), was soviel heißt wie »Zeilennummer nicht definiert«.

FAZIT

1. Wir haben bisher zwei Typen von Variablen kennengelernt:
   – den »numerischen« Variablen weisen wir ausschließlich Zahlenwerte zu
   – den Stringvariablen, gekennzeichnet durch das $-Zeichen am Ende des Variablennamens, weisen wir ausschließlich Zeichen, Buchstaben oder ganze Folgen von ihnen zu. Diese Zeichen- und Buchstabenfolgen heißen »Zeichenketten« oder »Strings«.
   Zahlen sind auch zugelassen, nur werden sie wie Zeichen (und nicht als Zahlenwerte) behandelt.
2. Die Zuordnung von Zahlen oder Strings zum falschen Variablentyp führt generell zu einer Fehlermeldung und oft auch zum Abbruch des Programms.
3. Das Semikolon am Ende eines PRINT-Befehls bewirkt, daß der nächste PRINT-Befehl, egal wo er steht, seinen Text direkt anschließend zum vorhergehenden druckt.
4. Um ein Programm ab einer bestimmten Zeilennummer zu starten, wird diese Zeilennummer dem RUN-Befehl angehängt (RUN 600).

Mit den String-Variablen haben wir jetzt das notwendige Handwerkszeug zur Erweiterung des INPUT-Befehls.

Wir können nämlich jetzt mit dem Befehl
INPUT A$
die Eingabe einer Information in Form eines Strings programmieren.

Bitte löschen Sie das letzte Programm mit NEW und geben Sie die beiden früheren INPUT-Programmzeilen neu ein und zwar jetzt so:

```
→ 20 INPUT A$      (RETURN)
  30 PRINT A$      (RETURN)
```

Wenn Sie nun RUN eingeben, wartet der Computer mit dem Fragezeichen, bis Sie einen String eingeben und mit RETURN abschließen. Dann erst druckt er den String aus.

In einem benutzerfreundlichen Programm sollte natürlich bei dem wartenden Fragezeichen von INPUT dabeistehen, welche Art von Eingabe erwartet wird. Der Benutzer des Programms sieht ja schließlich die Programmzeile nicht, in welcher der INPUT-Befehl – mit oder ohne »$« – steht.

Eine derartige Angabe können wir leicht erzeugen, indem wir eine entsprechende PRINT-Zeile vor den INPUT-Befehl der Zeile 20 setzen:

```
→ 10 PRINT"TEXT-EINGABE"
  20 INPUT A$
  30 PRINT A$
```

Als Ergebnis erhalten wir auf dem Bildschirm das Wort »TEXT-EINGABE«, darunter das Fragezeichen und wie gehabt den wartenden Cursor.

Wem das Untereinander nicht gefällt, der kann mit dem Semikolon als String-Kleber alles auf eine Zeile bringen:

```
→ 10 PRINT"TEXT-EINGABE";
```

Aber es geht noch viel eleganter!!

Der INPUT-Befehl selbst kann die Angabe enthalten. Wir nennen das einen »Kommentar« oder auf englisch einen »Prompt«.

```
→ 10
  20 INPUT"TEXT-EINGABE";A$
  30 PRINT A$
```

Zuerst wird die Zeile 10 gelöscht. Die Zeile 20 bringt dasselbe Ergebnis wie vorher die beiden Zeilen 10 und 20 zusammen.

Der Kommentar hinter dem INPUT muß immer zwischen Gänsefüßen stehen, darf maximal 38 Zeichen – auch Leerstellen zählen dazu – lang sein und muß von der Variablen (egal ob numerisch oder String) durch ein Semikolon getrennt sein.

Und noch eine feine Einrichtung hat der INPUT-Befehl: Man darf hinter ihn mehrere Variable hängen, die allerdings alle eingegeben werden müssen. Die Schreibweise ist der des PRINT-Befehls sehr ähnlich:

```
→ 40 INPUT"4 ZAHLEN";A,B,C,D
  50 PRINT A;B;C;D
  60 INPUT"3 STRINGS";A$,B$,C$
  70 PRINT A$ B$ C$
```

Sie sehen deutlich, daß der INPUT-Befehl ein Komma zur Trennung der Variablen verlangt, der PRINT-Befehl dagegen ein Semikolon, das aber, wie schon erwähnt, bei String-Variablen weggelassen werden kann.

Wenn Sie nun trotz des Kommentars, 4 Zahlen einzugeben, nur eine einzige eingeben und die RETURN-Taste drücken, gibt sich der Computer nicht zufrieden, sondern deutet mit einem doppelten Fragezeichen unmißverständlich darauf hin, daß noch weitere Eingaben erforderlich sind.

Die Zahlen, hintereinander eingegeben, müssen auch durch Kommata getrennt werden.

Basic-Befehl Nr. 6 INPUT

- darf nur innerhalb eines Programms (also nur mit Zeilennummer) eingesetzt werden
- wird immer in der Schreibweise: INPUT (Variable) verwendet
- druckt ein Fragezeichen auf den Bildschirm und wartet auf eine Eingabe von der Tastatur, die mit RETURN abgeschlossen werden muß
- akzeptiert numerische Variable (Zahlen) und String-Variable (Text)
- weist eine falsche Zuordnung zurück, d.h. eine Zahl kann nicht für eine String-Variable und ein String nicht für eine normale Variable eingegeben werden
- mit einem einzigen INPUT-Befehl kann die Eingabe von mehreren Variablen abgefragt werden. Diese Variablen müssen durch Kommata voneinander getrennt sein
- werden nicht alle geforderten Variablen eingegeben, fragt INPUT mit einem doppelten Fragezeichen nach den fehlenden Werten
- INPUT kann mit einem Kommentar versehen werden, den es vor dem Fragezeichen ausdruckt. Dieser Kommentar muß so geschrieben werden: INPUT "KOMMENTAR";(Variable)
- der Kommentar darf maximal 38 Zeichen lang sein

Der Sprungbefehl

Ich habe vor, als nächstes ein kleines, aber pfiffiges Spielprogramm mit Ihnen zu entwickeln. Doch dazu fehlen noch zwei weitere Basic-Befehle, auf die wir uns aber sofort stürzen wollen.

Die Sprache Basic enthält einen Befehl, der den Computer hüpfen läßt, und zwar auf eine mit diesem Befehl angegebene Programmzeile. Der Befehl lautet:

GOTO (Zeilennummer)

Er darf auch in der Form GO TO geschrieben werden.

Auf deutsch bedeutet er GEHE NACH.

Durch ihn veranlaßt, springt der Computer auf die angegebene Programmzeile und fährt dort mit dem Programm fort.

Der Befehl GOTO darf auch im Direkt-Modus verwendet werden.

Ich nehme an, daß Sie die letzte Version des INPUT-Programms noch im Computer haben. Wenn Sie jetzt direkt eintippen:

```
→ GOTO 60
```

springt der Computer auf die Zeile 60 und führt sie und die nachfolgende Zeile aus. Im Direkt-Modus wirkt er also wie der RUN-Befehl.

Im Programm-Modus sieht er genauso aus, er ist halt nur mit einer Zeilennummer versehen:

```
→ 35 GOTO 60
```

LISTen Sie bitte das vorhandene Programm erst noch einmal. Wir sehen jetzt:

```
20 INPUT "TEXT-EINGABE";A$
30 PRINT A$
35 GOTO 60
40 INPUT"4 ZAHLEN";A,B,C,D
50 PRINT A;B;C;D
60 INPUT"3 STRINGS";A$,B$,C$
70 PRINT A$ B$ C$
```

Durch den Sprungbefehl in Zeile 35 rückt das Programm auf die Zeile 60 vor und überspringt die Zeilen 40 und 50.

Der GOTO-Befehl kann als Zieladresse aber auch eine niedrigere Zeilennummer als er selbst haben. Dann springt er im Programm zurück und bildet so eine Schleife. Geben Sie bitte ein:

```
→ 80 GOTO 60
```

Dieser Sprungbefehl wiederholt die beiden Zeilen 60 und 70 endlos lange oder aber bis Sie die STOP- und die RESTORE-Taste drücken. Die STOP-Taste allein genügt nicht, weil sie vom INPUT-Befehl, der ja in der Schleife sitzt, nicht akzeptiert wird.

Basic-Befehl Nr.7 GOTO

- wird mit der folgenden Schreibweise verwendet: GOTO (Zeilennummer)
- veranlaßt den Computer sowohl im Direkt- als auch im Programm-Modus, auf die hinter dem Befehlswort stehende Zeilennummer zu springen;
- erlaubt Sprünge sowohl auf höhere, als auch auf niedrigere Zeilennummern; durch Rücksprünge können Programmschleifen erzeugt werden.

Wir können die obige Schleife ein bißchen verändern, indem wir die Zeile 80 so abändern

```
→ 80 GOTO 70
```

Jetzt springt der GOTO-Befehl immer wieder auf die Zeile 70 zurück, so daß in rasender Geschwindigkeit nur noch der PRINT-Befehl wiederholt und der Bildschirm mit dem Ausdruck der zuletzt eingegebenen Strings gefüllt wird. Zu bremsen ist dieser Vorgang diesmal mit der STOP-Taste. Der INPUT-Befehl, der vorher die STOP-Taste allein nicht akzeptiert hat und nur durch RESTORE gestoppt werden konnte, ist ja diesmal nicht in der Schleife drin.

Schleifen sind nicht zur Verzierung da

Schleifen sind nützliche Werkzeuge der Programmierung. Zum Beispiel können wir damit zählen. Das will ich Ihnen zeigen. Geben Sie ein:

```
→ NEW
  10 X = X + 1
  20 PRINT X
```

Nach RUN druckt uns die Zeile 20 eine 1 aus. Am Anfang des Programms ist der Variablen X noch keine Zahl zugeordnet, also ist sie 0. In Zeile 10 wird 1 dazugezählt, was 1 ergibt.

Das ist beileibe nicht trivial, denn jetzt ergänzen wir das Programm mit:

```
→ 40 GOTO 10
```

Die dadurch gebildete Schleife wiederholt den Vorgang, und in Zeile 10 wird bei jedem Durchgang die Variable X um 1 erhöht.

Dieses kurze Programm erzeugt eine Zählschleife, die eine endlose Zahlenkolonne über den Bildschirm sausen läßt. Mit der CTRL-Taste können Sie diesen Lauf verlangsamen, mit der STOP-Taste abbrechen.

Diese endlosen Schleifen sind aber wie Autobahnen ohne Ein- und Ausfahrt. Man kommt nicht drauf, und wer drauf ist, kommt nicht raus.

Wir brauchen also eine Methode, die den Aussprung aus einer Schleife ermöglicht. Dabei wollen wir aber angeben können, wann und wo wir ausspringen.

In einer Zählschleife wäre das denkbar beim Erreichen einer bestimmten Zahl, zum Beispiel 15.

Prüfung ohne Noten

Basic kennt einen Befehl, der prüft, ob eine bestimmte, von uns festlegbare Bedingung erfüllt ist. Dann nämlich tut er, was man ihm vorgegeben hat.

Der Befehl besteht aus den beiden Wörtern

IF THEN

auf deutsch WENN DANN.

Die volle Schreibweise sieht so aus:

```
IF (Prüfbedingung) THEN (Aktion)
IF       X = 3     THEN PRINT "DREI"
```

X=3 ist die Prüfbedingung, PRINT "DREI" die Aktion.

Der Befehl prüft also, ob die Prüfbedingung erfüllt ist oder nicht. Man sagt auch, er prüft, ob sie »wahr« oder »falsch« ist.

- ist sie erfüllt, dann wird die hinter dem THEN stehende Aktion ausgeführt
- ist sie nicht erfüllt, dann wird, ungeachtet dessen was noch in der Zeile steht, das Programm fortgesetzt.

Die Arbeitsweise sehen Sie am besten in unserem Beispiel. Fügen Sie bitte den Zeilen 10 und 20 die zwei folgenden Zeilen 30 und 50 hinzu:

```
→ 10 X = X + 1
  20 PRINT X
  30 IF X = 15 THEN GOTO 50
  40 GOTO 10
  50 PRINT "ENDE"
```

Also, in Zeile 10 wird X zu 1, Zeile 20 druckt die 1 aus.

Zeile 30 prüft, ob X bereits den Wert 15 erreicht hat. Dies ist nicht der Fall. Deswegen geht das Programm in der nächsten Zeile – nämlich 40 – weiter. Zeile 40 springt auf Zeile 10 zurück. X wird um 1 erhöht und hat jetzt den Wert 2. Die Bedingung in Zeile 30 ist aber noch immer nicht erfüllt. Also wiederholt sich der ganze Vorgang so lange, bis die Bedingung des IF-Befehls erfüllt ist. Erst wenn X den Wert 15 erreicht hat, tritt der THEN-Teil des Prüfbefehls in Kraft und führt das aus, was hinter dem THEN steht. In unserem Beispiel springt er mit GOTO 50 aus der Schleife heraus auf die Zeile 50.

In Zeile 50 endet das Programm mit dem Ausdruck »ENDE«.

Man muß bei der Festlegung der Prüfbedingung aufpassen, daß sie überhaupt auch auftritt. Wenn wir zum Beispiel in Zeile 10 den Wert für X jeweils um 2 erhöhen, also die Zeile 10 so schreiben:

```
→ 10 X = X + 2
```

ist die Schleife wieder endlos, weil 14 oder 16 erreicht wird, aber niemals 15.

Das Problem ist lösbar mit einer Prüfung, ob X größer als 15 geworden ist.

```
→ 30 IF X >  15 THEN GOTO 50.
```

Jetzt wird die Zeile erst bei der Zahl 16 verlassen.

Als Prüfbedingung stehen uns mehrere mathematische und logische Ausdrücke zur Verfügung.

ist gleich	=
kleiner	<
größer	>
kleiner oder gleich	<=
gleich oder größer	=>
ist ungleich	<>
Boole'sches UND	AND
Boole'sches ODER	OR
Negation	NOT

Die ersten sechs davon sprechen für sich selbst, die letzten drei (AND, OR UND NOT) werde ich in diesem Kurs nicht behandeln.

Mit der Ungleich-Bedingung können wir unsere Schleife sogar noch viel eleganter gestalten:

```
→ 30 IF X < > 15 THEN GOTO 10
  40 PRINT "ENDE"
  50
```

Wir sparen Zeile 50 ein, weil die Prüfung immer erfüllt ist, bis X den Wert 15 erreicht hat.

Die Prüf-Variable kann auch ein String sein. Bitte ergänzen Sie das Programm mit den folgenden Zeilen:

```
→  60 INPUT "JA ODER NEIN";A$
   70 IF A$ = "JA" THEN GOTO 100
   80 IF A$ = "NEIN" THEN GOTO 110
   90 PRINT "FALSCHE EINGABE":GOTO 60
  100 PRINT "JA"
  110 PRINT "NEIN"
```

Das wollen wir jetzt zeilenweise analysieren. Zeile 60 ist der uns schon bekannte INPUT-Befehl, der uns im Kommentar auffordert, entweder »JA« oder »NEIN« einzugeben.

Zeile 70 prüft, ob wir »JA« eingegeben haben. Ist das der Fall, dann springt sie auf Zeile 100, und wir erhalten den Ausdruck »JA«. Haben wir nicht das »JA« eingegeben, geht das Programm in der nächsten Zeile (80) weiter.

In der Zeile 80 wird geprüft, ob wir »NEIN« eingegeben haben. Wenn das zutrifft, springt sie auf Zeile 110, und das Wort »NEIN« wird ausgedruckt. Haben wir aber »NEIN« auch nicht eingegeben, sondern irgend ein anderes Wort (String), dann geht das Programm in der nächsten Zeile (90) weiter.

Ich hoffe, Sie sehen schon das typische Muster des IF..THEN-Befehls.

In Zeile 90 schließlich wird die Ermahnung ausgedruckt, doch nur mit »JA« oder »NEIN« zu antworten, und mit dem Rücksprung GOTO 60, mit dem Doppelpunkt als zweiter Befehl der Zeile 90 angehängt, wird eine neue Eingabe verlangt.

Einen kleinen Fehler hat das Programm noch. Wenn sie es mit RUN von Anfang an oder mit RUN 60 nur ab der INPUT-Zeile starten und gleich mit »JA« antworten, kommt zuerst Zeile 100 zum Zuge, aber gleich danach auch Zeile 110, und wir erhalten beide Ausdrucke, »JA« und »NEIN«.

Zeile 110 können wir in diesem Fall ausblenden, entweder durch eine Überbrückung mit GOTO (irgendwohin) oder mit

einem neuen Basic-Befehl, der das Programm mit Zeile 100 beendet. Er heißt END.

Endstation mit End

Ändern Sie bitte Zeile 100 ab!

```
→ 100 Print "JA" : END
```

Jetzt klappt es.

Die letzten beiden Angaben in der Befehlsbeschreibung des IF. .THEN-Befehls muß ich Ihnen noch vorstellen.

Wir dürfen nach einem IF. .THEN-Befehl als Aktion gleich noch einen IF. .THEN-Befehl anhängen. Dadurch sind mehrere Prüfungen möglich.

Ein Beispiel dafür könnte sein, daß im Dialog mit einem Programm nach »Wild« gefragt wird und nur die beiden Tiere »Reh« und »Hirsch« als richtige Antwort zugelassen sind – alle anderen Antworten werden abgewiesen.

Dieses Beispiel können Sie direkt an das bisherige Programm anhängen.

```
→ 120 INPUT "IHRE ANTWORT";A$
  130 IF A$ <> "REH" THEN IF A$ <> "HIRSCH"
  THEN 120
  140 PRINT "RICHTIG"
```

Zeile 120 bittet um eine Antwort, ohne zu sagen welche.

Zeile 130 prüft zuerst, ob »Reh« nicht die Antwort war. Ist das nicht der Fall (Vorsicht: doppelte Verneinung ist Bejahung), springt das Programm auf Zeile 140. Ist es der Fall, dann setzt Zeile 130 mit der zweiten Prüfung, nämlich ob die Antwort nicht »Hirsch« war, fort.

Auch jetzt springt das Programm bei Verneinung der Prüfung auf Zeile 140. Nur wenn beide Prüfungen richtig waren, nämlich weder »Reh« noch »Hirsch« eingegeben worden ist, dann tritt die Aktionsanweisung der Zeile 130 in Kraft, und das Programm springt zurück auf Zeile 120 und verlangt eine neue Antwort.

In Zeile 130 sehen Sie außerdem, daß ich nach dem THEN das GOTO weggelassen und gleich die Zeilennummer 120 geschrieben habe. Das ist erlaubt, aber, wie gesagt, nur bei der Kombination THEN GOTO.

Lassen Sie sich durch die »negative« Vorgehensweise dieses Beispiels nicht verunsichern. Mit einer »positiven« Frage geht es auch, nur wird dann das Programm länger.

Basic-Befehl Nr. 8 IF..THEN

- Nach dem Befehlswort IF steht eine Prüfbedingung, nach dem Befehlswort THEN eine Aktionsanweisung.
- IF prüft, ob die Prüfbedingung erfüllt ist: wenn ja, dann wird die Aktionsanweisung und der Rest der Zeile ausgeführt; wenn nein, dann läuft das Programm mit der nächsten Zeile weiter.
- Mehrere IF..THEN-Befehle können hintereinander gestellt eine Mehrfach-Prüfung bilden.
- Bei IF..THEN GOTO (Zeilennummer) kann das GOTO weggelassen werden.

Basic-Befehl Nr. 9 END

Dieser Befehl steht für sich allein oder nach einem IF..THEN-Befehl als Aktionsanweisung.
Er beendet sofort ein Programm.

Jetzt haben wir alles beisammen für ein erstes vollständiges Programm.

Wenn Sie den vorhergehenden Teil später nochmal durchgehen wollen, dann speichern Sie bitte das letzte Programm auf Band oder Diskette. Die Funktion der LOAD-, SAVE- und VERIFY-Befehle sind im Handbuch gut beschrieben. Wenn Sie zum erstenmal eine Diskette verwenden, denken Sie daran, daß Sie diese zuerst »formatieren«, das heißt, mit einem Namen und einer Identifizierungsnummer versehen müssen. Anleitung dazu gibt das Floppy-Handbuch.

Ein erstes Programm

Alle bisher behandelten Befehle kommen in einem kleinen Spielprogramm vor, welches ich dem Buch »Computerspiele und Knobeleien programmiert in Basic« von Rüdiger Baumann entnommen habe. Dieses Buch kann ich Ihnen sehr empfehlen, denn es ist keine Programmsammlung, sondern ein ausgezeichnetes Lehrbuch, mit leichten und anspruchsvollen Aufgaben.

Der Witz des Spiels beruht auf einem Trick mit den Zahlenwerten von Münzen. Herr Baumann nennt es »Welche Hand«. Ich erlaube mir, seine Aufgabenstellung zu zitieren:

»Der Computer fordert den Spieler auf, eine 10-Pfennig-Münze in die eine und eine 1-Pfennig-Münze in die andere Faust zu nehmen. Nun bittet er darum, den Wert der Münze in der RECHTEN HAND mit einer GERADEN ZAHL und den Wert der Münze in der LINKEN HAND mit einer UNGERADEN ZAHL zu multiplizieren, die Ergebnisse zu addieren und dem Computer mitzuteilen, ob die Summe GERADE oder UNGERADE ist.

Daraufhin teilt der Computer dem Spieler mit, in welcher Hand sich die 10-Pfennig-Münze und in welcher sich die 1-Pfennig-Münze befindet.«

Der Trick beruht auf den folgenden Regeln:

gerade * gerade = gerade
ungerade * gerade = gerade
ungerade * ungerade = ungerade
gerade + gerade = gerade
ungerade + gerade = ungerade

Wir haben nur zwei Fälle:

LINKS:

UNGERADE * 10 (PFG.) = GERADE
UNGERADE * 1 (PFG.) = UNGERADE

RECHTS:

GERADE * 10 (PFG.) = GERADE
GERADE * 1 (PFG.) = GERADE

Wir sehen, daß das Ergebnis in RECHTS immer eine gerade Zahl ist, egal mit welcher Münze. Also hängt die Summe aus RECHTS und LINKS nur vom linken Ergebnis ab:

Wenn die Summe ungerade ist, ist LINKS das 1-Pfennig-Stück, ist sie GERADE, dann ist links das 10-Pfennig Stück.

Diese Abfrage, Eingabe und Entscheidung wollen wir jetzt programmieren. Nehmen Sie bitte meine Zeilennummern, damit am Ende alles zusammenpaßt.

```
→ 300 INPUT "'GERADE' oder 'UNGERADE'";A$
  310 IF A$ = "GERADE" THEN xxx
  320 IF A$ = "UNGERADE" THEN yyy
  330 PRINT : PRINT
  340 PRINT "FALSCHE EINGABE" : GOTO 300
```

Diese 5 Zeilen sollten Ihnen jetzt schon vertraut sein.

Zeile 300 fordert zur Eingabe des Resultats der Kopfrechnung auf, mit einem entsprechenden Kommentar.

Zeile 310 prüft, ob die Antwort das Wort GERADE war. Ist das der Fall, springt sie nach weiter unten auf eine Zeilennummer, die wir noch nicht wissen. Ich habe sie deshalb xxx genannt; wir werden sie dann noch nachtragen.

War die Prüfung der Zeile 310 falsch, kommt die Zeile 320 an die Reihe mit der Prüfung auf das Wort UNGERADE. Auch deren Sprungadresse wissen wir noch nicht, und sie bekommt den vorläufigen Namen yyy.

Zeile 330 druckt uns aus Gründen der Lesbarkeit auf dem Bildschirm 2 Leerzeilen.

Zeile 340 schließlich kommt bekanntlich zum Zuge, wenn beide Prüfungen falsch waren. Sie mahnt zur richtigen Eingabe und springt zurück auf den INPUT-Befehl in Zeile 300.

Jetzt kommen die Konsequenzen aus den positiven Entscheidungen der Zeilen 310 und 320.

War das Resultat GERADE, dann ist Links das 10-Pfennig-Stück (siehe oben). Also definieren wir in der nächsten freien

Zeile –es ist 350 – eine String-Variable für das Zehnerl mit dem Namen M10$ und geben ihr den Wert LINKS

```
→ 350 M10$ = "LINKS"
```

Diese Zeile ist die Sprungadresse xxx in der Zeile 310, was wir dort nachtragen!

```
310 IF a$ = "GERADE" THEN 350
```

Wenn das Zehnerl links ist, muß rechts der Pfennig sein, also definieren wir entsprechend:

```
360 M01$ = "RECHTS"
```

Am Ende soll das Ergebnis ausgedruckt werden. Wir machen das ab Zeile 400.

```
→ 400 PRINT : PRINT
  410 PRINT "DIE 10-PFENNIG SIND ";M10$
  420 PRINT "DER 1-PFENNIG IST ";M01$
  430 END
```

Zeile 400 dient wieder der Schönschrift.

Zeile 410 setzt hinter den Text für die 10-Pfennig-Münze, mit einer Leerstelle und dem Semikolon-Kleber die String-Variable M10$.

Zeile 420 macht das gleiche für den Pfennig.

Zeile 430 ist nicht absolut notwendig, aber guter Brauch.

Jetzt fehlt uns nur noch der zweite Fall: wenn die Prüfung (in Zeile 320) eine Eingabe namens UNGERADE entdeckt hat. Jetzt ist, wie oben erläutert, das Zehnerl rechts und der Pfennig links, und die beiden String-Variablen M10$ und M01$ bekommen diese Werte zugewiesen:

```
→ 380 M10$ = "RECHTS"
  390 M01$ = "LINKS"
```

Nachzutragen ist der Wert für yyy in Zeile 320

```
→ 320 IF A$ = "UNGERADE" THEN 380
```

Probieren Sie es aus und lassen Sie es laufen. Wenn das Programm noch streikt, LISTen Sie es aus und suchen Sie den Schreibfehler, denn das ist es ja in den meisten Fällen. Zur Kontrolle können Sie auch in das separat abgedruckte komplette Programm schauen und eventuelle Dubiositäten beseitigen.

Das komplette Programm enthält, wie Sie sehen, noch die Anweisungen für das Spiel. Deren Erklärung kann ich mir schenken, denn es sind lediglich PRINT-Befehle, die für sich selbst sprechen.

Doch halt! Vier Dinge im Programm Nummer 1 kennen Sie vielleicht noch nicht.

Ich habe versucht, das Listing – so nennen wir den Ausdruck des Programms – lesbar anzulegen. Eine Methode dazu besteht darin, Leerzeichen an entsprechenden Stellen einzugeben. Basic macht nämlich keinen Unterschied zwischen:

PRINT A$ + B$ und PRINTA$+B$.

Die erste Version braucht zwar 3 Speicherplätze mehr – eben die 3 Leerzeichen, aber Version 2 ist schlecht lesbar.

Zur besseren Lesbarkeit gehört auch, daß ab und zu eine Leerzeile eingefügt wird, die den Programmablauf nicht beeinflußt. Im Programm sind dies die Zeilen 30, 270 und 290. Man erreicht dies durch einen Doppelpunkt nach der Zeilennummer, ohne Befehl.

Eine dritte Art, das Listing zu verbessern ist der Befehl REM.

Er kommt von REMARK und bedeutet »Bemerkung«. Mit diesem Befehl kann man Text in das Listing eingeben, der aber vom Programm völlig ignoriert wird. Im Beispiel sind dies die Zeilen 40 und 380.

Basic-Befehl Nr. 10 REM
Dieser Befehl erlaubt, einen Text oder Zeichen – mit Zeilennummer versehen – in den Ausdruck eines Programms (in ein Listing) einzufügen, ohne daß er im Ablauf des Programms erscheint. Es werden dabei keine Gänsefüße verwendet. Dieses Verfahren erhöht die Lesbarkeit des Listings.

Der vierte Punkt, der Ihnen im Listing des Programms Nummer 1 auffallen müßte, ist die Tatsache, daß bei allen PRINT-Befehlen der Text auf der linken Seite zwar mit einem Gänsefuß anfängt, aber rechts nicht mit dem Gänsefuß aufhört.

Das ist am Ende einer Zeile erlaubt, weil ja die Zeile mit der RETURN-Taste abgeschlossen worden ist.

FAZIT

1. Leerzeilen, die nur im Listing auftreten, das Programm aber nicht beeinflussen, erzeugt man durch einen Doppelpunkt hinter der Zeilennummer.
2. Bei der Abarbeitung einer Programmzeile ignoriert BASIC alle Leerzeichen, solange sie nicht zwischen Gänsefüßen stehend zu einem String gehören. Sie können daher weggelassen werden.
3. Am Ende einer Zeile kann der abschließende Gänsefuß weggelassen werden.

Es könnte sein, daß der eine oder andere Leser seufzt und meint, daß bei aller Kenntnis der Basic-Befehle das Ausdenken eines Programms viel zu schwer sei.

Deswegen wollen wir es ja üben, und ich habe aus derselben Quelle von R. Baumann, aus der auch Programm Nummer 1 stammt, eine andere Anwendung gefunden. Sie baut nicht auf einem so verflixten Trick auf, braucht aber drei weitere Basic-Befehle.

Der Zufall im Computer

Diese Überschrift ist eigentlich ein Widerspruch in sich, da doch im Computer alles durch die Anweisungen des Programms fest vorgegeben ist. Und doch gibt es etwas Zufallähnliches.

Es ist sowohl bei mathematischen Anwendungen als auch bei Spielen oft erforderlich, von einer nicht vorher bekannten, völlig zufällig entstandenen Zahl auszugehen. Wie könnte sie anders heißen als »Zufallszahl«.

Basic hat einen Befehl, der eine derartige Zufallszahl erzeugt. Er heißt:

RND(X)

Der Name ist abgeleitet aus RANDOM, was auf deutsch »zufällig« bedeutet.

Das X, das in Klammern hinter dem Befehlswort steht, wird Argument genannt. Das X kann drei Werte haben:

- eine positive Zahl (egal, welcher Wert)
- eine negative Zahl
- die Zahl 0

Bevor wir darauf eingehen, möchte ich Ihnen die Wirkungsweise von RND zeigen, wie immer am besten durch ein Experiment.

```
→ 10 A = RND(1)
  20 PRINT A
  30 GOTO 10
```

Diese drei Zeilen bilden eine Schleife, die Ihnen in einem Zahlenstreifen die Werte zeigt, die RND(1) erzeugt. Wir sehen vielstellige Zahlenwerte, die abgerundet zwischen 0.01 und 0.99 liegen.

Ab und zu taucht in der Zahlenkolonne eine Zahl in eigenartiger Schreibweise auf, mit einer Ziffer vor dem Dezimalpunkt und einem E – gefolgt von einer Zahl. Das ist die sogenannte wissenschaftliche Schreibweise, die im Handbuch von Commodore genauer erklärt wird. Hier bitte ich Sie, diese Zahlen ganz einfach zu ignorieren.

Zurück zu den Zahlen, die von RND erzeugt werden! Sie werden von einer Anfangszahl ausgehend durch eine komplizierte mathematische Formel errechnet, die zwar keine absolute Zufälligkeit garantiert, ihr aber sehr nahe kommt.

Sie können sich vorstellen, daß die Zufälligkeit ganz wesentlich von der oben genannten Anfangszahl abhängt.



Und gerade sie wird durch das Argument (X) beziehungsweise durch die drei Arten des Arguments bestimmt.

Das positive X beginnt nach dem Einschalten des Computers immer mit derselben Anfangszahl. Bei X = 0 wird als Anfangszahl der Stand der inneren Computeruhr genommen. Bei einer negativen Zahl ist diese Zahl der Anfangswert.

Falls Sie mehr über diese Methode wissen wollen, finden Sie eine Erklärung im 64'er Heft 8/85 auf Seite 131.

Ich empfehle Ihnen, den Wert 0 zu nehmen.

Mit 0 als Argument von RND will ich die drei Zeilen von oben neu schreiben, diesmal als sogenannten »Einzeiler«. Darunter versteht man ein Programm, welches in eine einzige Zeile paßt. Glauben Sie mir, in dieser Art sind schon ganze Kunstwerke veröffentlicht worden.

Diesen Anspruch erhebe ich nicht, nur den der Vereinfachung.

```
→ 10 PRINT RND(0) : GOTO 10
```

Die Zeile braucht unbedingt eine Zeilennummer, damit der GOTO-Befehl eine Schleife bilden kann.

Basic-Befehl Nr. 11 RND(X)
- wird für x die Zahl 0 oder irgendeine positive Zahl genommen, erzeugt dieser Befehl eine zirka achtstellige Zahl zwischen 0.01 und 0.99. Ihr jeweiliger Wert ist sozusagen zufällig.

Zufallszahlen haben wir jetzt, aber leider in der Form eines Dezimalbruches. Wenn wir in einem Programm zum Beispiel würfeln wollen, brauchen wir Zufallszahlen von 1 bis 6. Dezimalbrüche helfen uns da nicht viel. Wir brauchen also eine Möglichkeit, diese Dezimalbrüche in ganze Zahlen umzuwandeln.

Auch dafür hat Basic einen Befehl. Er lautet:

INT

Das ist die Abkürzung von INTEGER, was auf deutsch »ganze Zahl« bedeutet.

Der Dezimalbruch, der mit INT in eine ganze Zahl verwandelt werden soll, wird in Klammern hinter den Befehl geschrieben. Der Befehl ist im Direkt-Modus verwendbar:

```
→ PRINT INT(25.38) (RETURN-Taste)
```

Wir erhalten die Zahl 25.

INT macht es sich also leicht – es schneidet einfach alle Zahlen hinter dem Dezimalpunkt weg.

Es ist auch erlaubt, genau wie beim PRINT-Befehl eine Formel mit INT zu versehen:

```
→ PRINT INT(2.8908*567.8)
```

Das Ergebnis ist 1641.

Basic-Befehl Nr. 12 INT(A)
- wandelt einen Dezimalbruch A in eine ganze Zahl um.
- A kann nicht nur ein Dezimalbruch, sondern eine mathematische Formel oder ein komplizierter mathematischer Ausdruck sein.
- Die Klammern dürfen nicht weggelassen werden.

Diesen Befehl INT wenden wir jetzt an, um aus den unerwünschten Dezimalbrüchen des RND-Befehls ganze Zahlen zu machen.

Zuerst nehme ich der Klarheit halber die 3-zeilige Version, welche die Zufallszahl einer Variablen zuordnet:

```
→ 10 A = RND(0)
  20 PRINT INT(A)
  30 GOTO 10
```

Der Unterschied zu vorher liegt hier in der Zeile 20. Das Resultat nach RUN ist enttäuschend, aber verständlich, denn der INT-Befehl macht aus Dezimalbrüchen, die kleiner als 0 sind, nur eine 0.

Zeile 20 muß verbessert werden, indem der Dezimalpunkt von A nach rechts verschoben wird. Das erreichen wir durch die Multiplikation mit 10, 100, 1000 und so weiter.

```
→ 20 PRINT INT(A*10)
```

liefert Zufallszahlen von 0 bis 9. Den Bereich 0 bis 99 erhalten wir durch:

```
→ 20 PRINT INT(A*100)
```

Als Einzeiler sieht diese letzte Version so aus:

```
→ 10 PRINT INT(RND(0)*100) : GOTO 10
```

Beachten Sie bitte die Verschachtelung der Klammern:
INT(RND(0))
1. Klammer auf 1. Klammer zu
2. Klammer auf 2. Klammer zu

Wenn eine Klammer fehlt oder falsch ist, erhalten wir die Fehlermeldung »SYNTAX ERROR«. Am besten ist es, Sie überprüfen, daß die Anzahl der geöffneten und geschlossenen Klammern immer gleich ist.

Wählbarer Bereich der Zufallszahlen

Ich gebe Ihnen ein Kochrezept, mit dem Sie Zufallszahlen innerhalb des von Ihnen gewünschten Bereiches erzeugen können.

Die Formel dazu lautet:

```
→ A = INT(RND(0)*X)+Y
```

Sie erzeugt ganze Zahlen innerhalb des Zahlenbereiches Y bis Y + X - 1.

Beispiel:

Zufallszahlen von 20 bis 90 machen Y zu 20 und erfordern ein X, das aus der Formel Y + X - 1 = 90 errechnet wird. Anders geschrieben lautet diese Formel X = 90 - Y + 1. Das ergibt in unserem Fall X = 71.

```
→ 10 PRINT INT(RND(0)*71)+20
```

Im Zweifelsfall lohnt es sich immer, die Formel mit einer derartigen Zeile 10 auszuprobieren.

Noch eine Eingabe per Tastatur

Die Eingabe von Daten über die Tastatur in ein Programm haben wir bislang mit dem Befehl INPUT schon ausführlich geübt.

In Basic gibt es noch einen zweiten Befehl, der im Prinzip das gleiche macht. Nur im Detail der Ausführung unterscheidet er sich vom Kollegen INPUT. Diese Unterschiede aber machen ihn zu einer wertvollen Alternative.

Der Befehl heißt

GET (Variable)

was soviel bedeutet, wie »holen, bekommen«.

Der GET-Befehl holt also einen Wert und weist ihn der hinter ihm stehenden Variablen zu. Diese Variable kann eine numerische oder eine String-Variable sein.

Unterschied zwischen GET UND INPUT
- INPUT wartet, bis eine Eingabe mit der RETURN-Taste abgeschlossen ist
- GET wartet nicht, sondern prüft lediglich, ob eine Taste gedrückt worden ist.
- INPUT erlaubt die Eingabe bis zu 78 Zeichen
- GET holt immer nur ein Zeichen
- INPUT meldet sich mit Fragezeichen und Cursor
- GET meldet sich auf dem Bildschirm nicht

Die Frage stellt sich, wie der GET-Befehl prüfen kann, ob eine Taste gedrückt worden ist, wenn er nicht wartet. Daß er gerade in dem kurzen Zeitpunkt prüft, in dem zufällig der Tastendruck stattfindet, ist mehr als unwahrscheinlich.

Und in der Tat – wenn nicht das Zeichen jeder gedrückten Taste in einen Pufferspeicher käme, ehe es weiterverwendet wird, hätte der GET-Befehl keine Chance, jemals eine Eingabe zu erwischen.

Der Tastaturpuffer

Diesen Speicher des Computers, in dem die Zeichen gedrückter Tasten erst einmal zwischengelagert werden, möchte ich Ihnen kurz zeigen.



Dazu brauchen wir eine Zählschleife, wie wir sie beim GOTO-Befehl erstmals angewendet haben. Hier brauche ich sie, um eine sogenannte Zeitverzögerungsschleife zu bauen. Diese tut das, was ihr Name sagt: sie zählt eine gewisse Zeit still vor sich hin und verzögert so den nächsten Programmschritt. Entfernen Sie bitte alle Programmzeilen mit NEW

```
→ NEW
  30 X = X + 1
  40 IF X <> 500 THEN 30
```

Mit RUN gestartet, läuft die Schleife in diesen beiden Zeilen 500 mal durch, bis sich der Editor mit READY und Cursor wieder meldet.

Diese Zeit nützen wir, um so viele Tasten wie möglich hintereinander zu drücken. Natürlich sehen wir nichts auf dem Bildschirm, während das Programm der Zeitschleife läuft. Die Behauptung, daß die gedrückten Zeichen in einen Pufferspeicher kommen, bewahrheitet sich nach dem Ende der Zeitschleife.

Nach dem READY holt nämlich der Editor alle Zeichen aus dem Tastaturpuffer und druckt sie aus.

Da dieser Puffer nur 10 Zeichen speichern kann, sehen Sie nur 10 Zeichen, selbst wenn Sie es geschafft haben, mehr einzugeben.

Also nochmal:

→ RUN eingeben
möglichst viele Tasten drücken, aber nicht gleichzeitig
nach dem Ready die Zeichen zählen

In diesem Tastaturpuffer schaut also der GET-Befehl nach, ob vorher ein Tastendruck ein Zeichen darin untergebracht hat. Wenn ja, dann verwendet er es, falls nein, kommt die nächste Programmzeile dran.

Um das zu zeigen, erweitern wir das obige Programm um vier weitere Zeilen:

```
→ 10 GET A
  20 PRINT A
  30 X = X + 1
  40 IF X 100 THEN 30
  50 X = 0
  60 GOTO 10
```


In Zeile 10 schaut der GET-Befehl im Tastaturpuffer nach einem Zeichen. Was immer er findet, druckt Zeile 20 auf den Bildschirm. Eine 0 signalisiert, daß der Puffer leer war.

Zeile 30 und 40 zählt von 1 bis 100, wie beim Versteckspielen, und macht dann nach 100 Zähleinheiten in Zeile 50 weiter, in der die Zählvariable X wieder auf Null gesetzt wird.

Zeile 60 springt zurück auf den GET-Befehl und alles fängt mit X = 0 wieder von vorn an.

Das Resultat ist eine langsam fortschreitende senkrechte Kolonne von Ziffern links am Bildschirm. Die Geschwindigkeit ist durch die Prüfzahl in Zeile 40 einstellbar.

Während der Zählschleife haben wir nun Gelegenheit, Zahlen einzugeben, die als einzelne Ziffern vom GET-Befehl geholt und durch Zeile 20 ausgedruckt werden.

Wenn Sie einen Buchstaben eingeben, bricht das Programm mit einer Fehlermeldung ab.

Um Buchstaben und Zeichen eingeben zu können, müssen wir die numerische Variable A in eine Stringvariable A$ umwandeln – natürlich in beiden Zeilen 10 und 20.

Jetzt holt der GET-Befehl Buchstaben – aber auch Ziffern! Diese werden jedoch als Zeichen (einstellige Strings) und nicht als Zahlenwerte behandelt.

Basic-Befehl Nr. 13 GET

- Der Befehl GET (Variable) holt eine Zahl oder ein Zeichen aus dem Tastaturpuffer und weist es der Variablen zu.
- Die Variable kann eine numerische oder eine String-Variable sein.
- Entspricht das Zeichen beziehungsweise die Zahl nicht dem Variablentyp, wird mit Fehlermeldung abgebrochen.
- GET wartet nicht. Falls der Puffer leer ist, weist er der Variablen den Wert 0 beziehungsweise einen sogenannten »Nullstring« (kein Leerzeichen, einfach gar nichts) zu.

Jetzt will ich Ihnen eine echte Anwendung des GET-Befehls zeigen.

Wir haben im Kapitel »Prüfung ohne Noten« einen Programmteil geschrieben, um den Benutzer zu fragen, ob er das Programm noch einmal laufen lassen möchte oder nicht.

Mit dem INPUT-Befehl ging das so:

```
→ 10 PRINT "BEGINN"
   .
   .    (Programm)
   .
  320 INPUT "NOCH EINMAL (J/N)";A$
  340 IF A$ = "J" THEN 10
  350 IF A$ = <>"N" THEN 320
  360 PRINT "AUF WIEDERSEHEN"
```

Ab Zeile 10 beginnt das Programm.
Zeile 320 fragt nach J(a) oder N(ein).
Zeile 340 prüft, ob die Eingabe "J(a)" ist, im Fall »wahr« springt das Programm an den Anfang zurück.

Im Fall »falsch« kommt Zeile 350 an die Reihe mit der Prüfung, ob die Eingabe ein ungültiges Zeichen ist – alle Zeichen außer »NEIN« sind ungültig. Leider ist die Eingabe »NEIN« auch falsch, denn wir prüfen ja nur auf »N«. Das ist ein Nachteil des INPUT-Befehls.

Im Fall »wahr« wird der INPUT-Befehl wiederholt. Wenn es ein »N« war, verabschiedet sich das Programm in Zeile 360.

Dasselbe machen wir jetzt mit dem GET-Befehl. Die Zeilen 10, 340, 350 und 360 bleiben unverändert.

```
→ 10 PRINT "BEGINN"
   .
   .    (PROGRAMM)
   .
  320 PRINT "NOCH EINMAL (J/N)?"
  330 GET A$: IF A$ = "" THEN 330
  340 IF A$ = "J" THEN 10
  350 IF A$ <>"N" THEN 320
  360 PRINT "AUF WIEDERSEHEN"
```

Zeile 320 druckt den Prompt aus. In Zeile 330 schaut der GET-Befehl im Tastaturpuffer nach, ob er ein Zeichen enthält. Mit IF A$ = " " prüft er, ob der Puffer leer ist. Die doppelten Gänsefüße ohne Zwischenraum repräsentieren den bei der Befehlsbeschreibung genannten »Nullstring«, das heißt, im Puffer war kein Zeichen. Deswegen springt diese Zeile 330 auf sich selbst zurück und verharrt in dieser Schleife so lange, bis die Prüfbedingung des Nullstrings nicht mehr erfüllt ist. Das ist sie, sobald irgendeine Taste gedrückt worden ist.

Die restlichen Zeilen prüfen genau so wie im Beispiel vorher.

Der Hauptunterschied ist der, daß der GET-Befehl auch »NEIN« akzeptiert, prüft er doch immer nur 1 Zeichen, nämlich das erste – er würde natürlich auch »NAME« akzeptieren. Außerdem reagiert er sofort und nicht erst nach dem Drücken der RETURN-Taste.

Prinzipiell möchte ich sagen, daß der GET-Befehl »intelligenter« ist als der INPUT. Er bietet mehr Flexibilität und auch mehr Eleganz – eine Eigenschaft, in der sich oft gute und schlechte Programme unterscheiden.

Wir sind gerüstet für ein zweites Programm. Es heißt »ZAHLEN RATEN« und ist ebenfalls aus dem Buch »Computerspiele und Knobeleien« von Rüdeger Baumann abgeleitet.

Aufgabe:

Eine vom Computer zufällig gewählte Zahl zwischen 1 und 100 soll in möglichst wenigen Versuchen erraten werden. Nach jedem Rateversuch gibt der Computer einen Hinweis »zu groß« oder »zu klein«. Wurde richtig geraten, nennt der



Computer die Zahl der Versuche und stellt anheim, das Spiel noch einmal zu versuchen oder zu beenden.

Die Planung eines Programms

Ich habe ganz am Anfang dieses Kurses erwähnt, daß Basic dazu verleitet zu »hacken«. In der Tat führt die an sich sehr positive Eigenschaft dieser Programmiersprache, nämlich schon kleinste Bruchstücke eines Programms eingeben und dann gleich ausprobieren zu können, leicht zu einem sogenannten »Spaghetti-Programm«. Die einzelnen Bruchstücke kann man nämlich stehen lassen und die anderen Teile irgendwie dazwischenschieben oder mit GOTO irgendwo anhängen. Das Resultat ist aber am Schluß ein wilder Haufen von Befehlszeilen, die zwar funktionieren, aber nicht mehr nachvollziehbar sind.

Es ist daher sehr zu raten, sich den Ablauf eines Programms vorher zu überlegen, was ja nicht schwer ist, weil ein Computer nie mehrere Sachen gleichzeitig macht, sondern alle streng hintereinander abwickelt. So einen Ablauf kann man schön aufs Papier malen, und das will ich Ihnen jetzt zeigen.

Zuerst schreiben wir ganz einfach alle Vorgänge der Reihe nach auf. Wenn die Reihenfolge uns nicht paßt, wird sie einfach abgeändert.

Am Ende kann es so aussehen:

1. Bildschirm löschen
2. Überschrift schreiben
3. Zufallszahl zwischen 1 und 100 erzeugen
4. Zähler für Zahl der Versuche auf Null stellen
5. Anweisungen an den Spieler
6. Eingabe einer geratenen Zahl
7. Zähler der Versuche um 1 erhöhen
8. Prüfung der geratenen Zahl auf:
   - zu groß: Hinweis geben und neuer Versuch
   - zu klein: Hinweis geben und neuer Versuch
   - richtig: Zahl der Versuche ausdrucken
9. Frage ob noch einmal
   - ja: zurück zum Anfang
   - nein: Verabschiedung, Ende

Im Bild 3 habe ich diese 9 Ereignisse praktisch genauso genommen, wie sie hier stehen, und habe sie in ein sogenanntes »Fluß-Diagramm« eingezeichnet, in welchem durch Linien der Verlauf, insbesondere der Verzweigungen und Schleifen besonders gut sichtbar wird.

Alles was jetzt noch zu tun bleibt, ist, die einzelnen Kästchen mit den Befehlen, die wir kennen, als Programmzeilen zu schreiben. Dabei können wir im Fluß-Diagramm unsere Rücksprungadressen und andere Notizen eintragen.

Übrigens: ich schreibe hier nur die »aktiven« Programmzeilen. Alle anderen Teile des Programms, die nur der Lesbarkeit dienen, entnehmen Sie bitte dem nebenstehenden kompletten Listing des Programms.

Punkt 1 und 2:

Den Bildschirm löschen wir mit einem PRINT-Befehl, hinter dem wir nach dem ersten Gänsefuß die geSHIFTete CLR/HOME-Taste drücken – was bekanntlich im Direkt-Modus den Löschvorgang auslöst. Im Listing erscheint dann ein invertiertes Herz. Was es damit auf sich hat, erkläre ich im Anschluß an das Programm.

```
→ 10 PRINT" (SHIFT CLR/HOME)"
  20 PRINT"****** ZAHLEN RATEN******
```

Punkt 3:

Der Bereich der Zufallszahl soll zwischen 1 und 100 liegen. Entsprechend dem dafür angegebenen Kochrezept INT(RND(0)*X)+Y errechnen wir wieder mit der unteren Grenze 1=Y und der oberen Grenze X+Y-1=100 das uns noch fehlende X:

Bildschirm löschen

Überschrift

Zufallszahl zwischen 1 und 100 erzeugen

Zähler für Zahl der Versuche auf Null

Anweisungen an den Spieler

Eingabe einer geratenen Zahl

Zähler der Versuche um 1 erhöhen

Prüfung

zu groß | richtig | zu klein

Zahl der Versuche ausdrucken

Frage ob noch einmal

ja | nein

Verabschiedung Ende

Bild 3. Ein Programm-Ablaufplan

X+1-1=100
X=100

Dann ergibt sich für die Zufallszahl »Z« die folgende Programmzeile:

```
→ 70 Z=INT(RND(0)*100)+1
```

Punkt 4:

Der Zähl-Variablen für die Anzahl der Versuche geben wir den Namen »V«. In Zeile 80 wird sie auf Null gesetzt.

```
→ 80 V=0
```

Punkt 5:

Die Anweisungen an den Spieler stehen in Zeile 120 und 130 (siehe Listing).

Sie können die Anweisung aber Ihrem Geschmack entsprechend auch anders schreiben oder anders anordnen.

Punkt 6:

Die Aufforderung zur Eingabe einer Zahl und die Eingabe selbst besorgt uns wieder der INPUT-Befehl

```
→ 140 PRINT
  150 INPUT"WELCHE ZAHL IST ES";R
```

Vor dem INPUT-Befehl drucken wir erst eine Leerzeile aus, damit es besser aussieht.

Der einzugebenden Zahl stellen wir die numerische Variable »R« zur Verfügung, diesmal ohne $-Zeichen, da wir ja Zahlen eingeben wollen.

Punkt 7:

Sobald die Zahl eingegeben ist, muß die Variable V des Versuche-Zählers um 1 erhöht werden.

```
→ 160 V=V+1
```

Punkt 8:

Jetzt kommt die Prüfung mit IF-THEN. Ich habe sie wegen der nachfolgenden Hinweise ZU GROSS, ZU KLEIN und RICHTIG in 3 Gruppen zu je 2 Stufen aufgebaut.

Das Grundprinzip dieser Prüfung liegt im Vergleich der beiden Zahlen, nämlich der geheimen Zahl des Rechners »Z« und der eingegebenen Zahl »R« des Raters. Zeile 200 prüft auf »größer«, Zeile 210 auf »kleiner«, Zeile 220 auf »gleich«.

```
→ 200 IF R > Z THEN PRINT R;:GOTO xxx
  xxx PRINT"IST ZU GROSS":GOTO 140
```

Die Zeilennummer xxx tragen wir dann nach, sobald klar ist, wieviele Zeilen noch dazwischen liegen.

Wichtig in Zeile 200 ist das Semikolon nach der geratenen Zahl R. Es klebt wieder einmal den nachfolgenden Satz der Zeile xxx direkt hinter den Wert der Zahl R, so daß auf dem Bildschirm der Satz erscheint:

»(Zahl R) IST ZU GROSS«.

Genau so ist es mit den beiden anderen Prüfungen:

```
→ 210 IF R < Z THEN PRINT R;:GOTO yyy
  yyy PRINT"IST ZU KLEIN":GOTO 140
  220 IF R = Z THEN PRINT R;:GOTO zzz
  zzz PRINT"IST RICHTIG"
```

Aus der Reihenfolge wird jetzt klar, daß xxx die Zeile 230 ist, entsprechend yyy=240 und zzz=250. Bitte tragen Sie dies nach.

Die beiden falschen Ergebnisse in Zeile 230 und 240 führen also zum Rücksprung auf eine neue Eingabe, die ab Zeile 140 beginnt.

Ein richtiges Ergebnis führt in die nächsthöhere Zeile 260 weiter.

In Zeile 260 und 270 drucken wir jetzt den letzten Stand des Versuche-Zählers V aus, eingebettet in Text, von dem er durch zwei Semikolons getrennt werden muß:

```
→ 260 PRINT"SIE HABEN";V;"VERSUCHE ";
  270 PRINT"BENOETIGT"
```

Ich habe mit Absicht den Text, der ja leicht in eine Zeile gepaßt hätte, in zwei Zeilen getrennt. Erstens verbessert er das Format des Listings. Zweitens aber wollte ich Ihnen zeigen, wie man trotzdem den Text in eine Zeile ausgedruckt erhält. Das Geheimnis liegt wieder im Semikolon nach dem letzten Wort der Zeile 260; zusätzlich aber ist die Leerstelle vor dem abschließenden Gänsefuß wichtig. Ohne sie würden die beiden Wörter »VERSUCHE« und »BENOETIGT« zu einem einzigen Wort verklebt werden.

Punkt 9:

Es bleibt noch die Frage nach einer Wiederholung. Das machen wir mit dem GET-Befehl, genau wie vorher schon.

Zeile 320 stellt die Frage, in Zeile 330 wartet der GET-Befehl mit einer Nullstring-Schleife auf eine gedrückte Taste. Zeile 340 prüft auf »J(a)« und springt zurück auf den Anfang, Zeile 350 prüft auf ungültige Eingaben mit Wiederholung der Aufforderung, und wenn »N(ein)« eingegeben ist, schließt Zeile 360 das Programm mit höflicher Verabschiedung.

```
→ 320 PRINT"NOCH EINMAL (J/N) ?"
  330 GET A$:IF A$="" THEN 330
  340 IF A$="J"THEN 10
  350 IF A$ <> "N" THEN 320
  360 PRINT:PRINT:PRINT"AUF WIEDERSEHEN"
  370 END
```

So, das war doch nicht schwer, oder?

Alles, was passieren kann, sind Tippfehler beim Eingeben.

Programmierte Steuertasten oder der Gänsefuß-Modus

In Zeile 10 des 2. Programms »Zahlen Raten« kommt ein invertiertes Zeichen in Gänsefüßen vor. Es ist das Symbol der Steuertaste CLR, eingebettet in ein Programm.

Was bedeutet das ?

Der Computer nützt hier eine Eigenschaft des PRINT-Befehls aus.

PRINT druckt bekanntlich sowohl im Direkt-, als auch im Programm-Modus alle Zeichen auf den Bildschirm, die in Gänsefüßen hinter ihm stehen.

```
→ PRINT"AAABBB"
```

druckt nach RETURN diese 6 Buchstaben aus.

Was ist aber mit den Steuertasten CLR, HOME INST, DEL CRSR rauf/runter/links/rechts, und was ist mit den Farbtasten?

Versuchen Sie es! Sie werden sehen, daß auch diese Tasten, zwischen Gänsefüßen, ein Symbol auf dem Bildschirm erzeugen, und zwar immer invertiert.

In einem PRINT-Befehl erscheinen diese invertierten Zeichen nur im Listing.

Bei der Ausführung des PRINT-Befehls wird ihre Funktion ausgeführt!!

Das müssen Sie ausprobieren. Geben Sie einen PRINT-Befehl mit Buchstaben, gefolgt von der CTRL-Taste gleichzeitig gedrückt mit der 8-Taste (yel = yellow = gelb) und dann wieder Buchstaben ein.

```
→ PRINT"AAA (CTRL 8) BBB"
```

Statt (CTRL 8) sehen Sie auf dem Bildschirm ein invertiertes Pi.

Sobald Sie jetzt RETURN drücken, werden die B und alles folgende in Gelb gedruckt. Wenn Sie die ursprüngliche Farbe wiederherstellen wollen, drücken Sie entweder auf die Commodore-Taste (C=) und die 7 gleichzeitig oder aber auf STOP und RESTORE.

Genau so wie mit den Farb-Tasten geht es mit den Cursor-Tasten. Zur Abwechslung machen wir das im Programm-Modus.

```
→ 10 PRINT"ZZZ (4mal CRSR rechts) XXX"
```

Nach RUN werden die Z noch normal, die X aber um 4 Stellen nach rechts versetzt gedruckt.

Die folgende Tabelle 1 auf Seite 63 zeigt Ihnen die invertierten Steuerzeichen zur leichteren Identifzierung mit der Funktionsbeschreibung.

Es gibt noch einige Funktionen, die ihre eigenen invertierten Steuerzeichen haben, aber keine eigenen Tasten dafür. Weil das schon in die höhere Kunst der Programmierung reicht, verweise ich hier nur auf das 64'er Sonderheft 2/86. Da sind sie auf den Seiten 36 und 40 beschrieben.

Der ASCII-Code

Ganz rechts in der Tabelle der Steuertasten ist eine Spalte von Zahlen, die unter der Überschrift ASCII stehen. Was dahinter steckt, sei kurz beschrieben.

Alle Computer verwenden intern irgendwelche Code-Zahlen, um die Zeichen, Buchstaben und Zahlen im Rechenwerk, im Speicher und in den anderen Einheiten des Computers darzustellen.

Theoretisch kann das jeder Computerhersteller machen, wie er will. Nur, wenn Daten von einem Gerät an ein anderes geliefert werden, zum Beispiel vom Computer an einen Drucker, müssen die Daten einem international standardisierten Code entsprechen.

Dieser Standard heißt »American Standard Code for Information Interchange«, abgekürzt ASCII.

Jedes Zeichen, jede Zahl und jede Funktion hat seinen eigenen Code-Wert. Sie sind (fast) alle in einer Tabelle im Anhang F des Commodore-Handbuches aufgelistet. In der nebenstehenden Tabelle sind ganz einfach diese Werte nur für die Steuertasten ausgewählt und dargestellt.

Wir können aber ganz leicht ein kleines Programm schreiben, das uns die Abfrage aller ASCII-Codes gestattet. Dazu muß ich allerdings einen Befehl vorwegnehmen, den ich ein paar Absätze weiter unten erkläre. Aber das macht nichts - Sie können mir sicher folgen.

```
→ 10 INPUT A$
  20 A=ASC(A$)
  30 PRINT A
  40 GOTO 10
```

Der Pfiff liegt natürlich in dem neuen Befehl ASC(A$) in Zeile 20. Er wandelt den ersten Buchstaben des per INPUT eingegebenen und mit RETURN abgeschlossenen Strings in seinen ASCII-Codewert um, den wir in Zeile 30 ausdrucken.

Zeile 40 verschafft dem Programm die Wiederholbarkeit, die bei INPUT nur durch die STOP- und RESTORE-Taste erreicht werden kann.



Jetzt können Sie nach Herzenslust alle Tasten und Tastenkombinationen nach ihrem ASCII-Code abfragen und so die unvollständige Liste im Commodore Handbuch vervollständigen.

Befehle, die Strings verändern

Da mit dem PRINT-Befehl ja auch Daten an ein anderes Gerät gegeben werden, nämlich an den Bildschirm, versteht er die ASCII-Werte auch.

Der Tabelle im Handbuch entnehmen wir zum Beispiel den ASCII-Wert für das A, er ist 65. Um mit dieser Zahl den Buchstaben A auf den Bildschirm zu zaubern, gibt es einen weiteren BASIC-Befehl:

`CHR$(ASCII-Wert)`

Er ist eine Abkürzung von »Character«, was auf deutsch »Zeichen« bedeutet. Das String-Symbol muß immer dahinter stehen.

Basic-Befehl Nr. 14 CHR$(X)

- wandelt die ASCII-Codezahl X in ihr entsprechendes Zeichen um.
- Die Zuordnung der ASCII-Codes und der Zeichen ist im Handbuch als Anhang enthalten.

In Verbindung mit dem PRINT-Befehl sieht der CHR$-Befehl so aus:

```
→ PRINT CHR$(65)
```

Diese Zeile, sowohl im Direkt-Modus als auch in einer Programmzeile, bewirkt dasselbe wie

PRINT"A".

Sie druckt den Buchstaben A auf den Bildschirm.

Ja mei, werden Sie sagen. Was bringt denn das? Die gute alte Gänsefuß-Methode ist doch viel praktischer und kürzer. Aber die Methode der ASCII-Codewerte kann dennoch recht vielseitig sein.

Mit Zahlen, auch mit Codezahlen, kann man nämlich rechnen. Geben Sie bitte folgende Zeilen ein:

```
→ 10 X=64
  20 X=X+1
  30 PRINT CHR$(X);
  40 IF X=90 THEN END
  50 GOTO 20
```

In Zeile 10 geben wir der numerischen Variablen X den Wert 64, das ist um 1 weniger als die Codezahl des Buchstaben A. In Zeile 20 wird eine Zählschleife begonnen mit der Erhöhung von X um 1. Zeile 30 druckt das dieser Zahl entsprechende Zeichen aus. Im ersten Durchlauf ist es das A.

Zeile 50 schließt die Zählschleife durch den Rücksprung auf Zeile 20. Dadurch werden alle Zeichen vom Codewert 65 bis 90 - das ist das Alphabet von A bis Z - ausgedruckt.

Die Prüfzeile 40 wartet bis das Z den Wert 90 erreicht hat und beendet dann das Programm.

Nur mit Buchstaben in Gänsefüßen wäre dieses Programm recht lang geworden.

Der Befehl CHR$ wird hauptsächlich bei der Verarbeitung von Strings, also bei Erkennen, Vergleichen und Verändern von Wörtern eingesetzt. Er gehört zu einer ganzen Gruppe von Befehlen, die alle mit der String-Verarbeitung zu tun haben. Wir werden sie alle noch kennenlernen.

Der direkte Nachbar zu CHR$ ist der Befehl

`ASC (String)`

Er ist die exakte Umkehrung des CHR$-Befehls. ASC ist die Abkürzung von ASCII.

Erinnern Sie sich, ich habe diesen Befehl schon kurz verwandt, ein paar Absätze früher bei der Vorführung des ASCII-Codes.

Der Befehl wandelt das erste Zeichen des in Klammern stehenden Strings in seinen ASCII-Codewert um.

```
→ PRINT ASC("WORT")
```

Diese Zeile druckt den ASCII-Wert von W aus, es ist 87.

Basic-Befehl Nr.15 ASC(String)
wandelt das erste Zeichen des Strings in seinen ASCII-Codewert um.

Als Beispiel verwende ich die schon mehrfach geübte Frage nach Wiederholung oder Ende eines Programms, diesmal aber durch Vergleich der ASCII-Werte.

```
→ 10 PRINT"PROGRAMM-ANFANG"
   .
   .     (Programm)
   .
  100 PRINT"NOCH EINMAL (J/N) ?"
  110 GET A$:IF A$="" THEN 110
  120 X=ASC(A$)
  130 IF X=74 THEN 10
  140 IF X <> 78 THEN 100
  150 END
```

In Zeile 10 soll das Programm beginnen – welches, interessiert uns hier nicht weiter. Am Ende dieses Programms fordert Zeile 100 zur Eingabe von JA oder NEIN auf. Zeile 110 enthält das gewohnte Bild der GET-Schleife, die wartet, bis ein Zeichen im Tastaturpuffer erscheint.

Zeile 120 wandelt das erste Zeichen des eingegebenen Wortes in seinen ASCII-Code um. Aus der ASCII-Codetabelle wissen wir, daß dem erwarteten J die Zahl 74, dem N aber die Zahl 78 entspricht. Alle anderen Werte sollen zurückgewiesen werden.

Ist A=74, dann springt Zeile 130 auf den Anfang des Programms zurück.

Ist A nicht 78, wird durch Zeile 140 die Eingabe wiederholt.

Im Fall, daß A=78, fällt die Prüfung in Zeile 140 durch, und das Programm macht in Zeile 150 Schluß.

Auch dieses Programm ist länger als die alte Methode, aber es zeigt wenigstens die Arbeitsweise von ASC.

Der Befehl ASC hat den Nachteil, daß er immer nur das erste Zeichen des Strings zur Umwandlung nimmt. Um ein ganzes Wort in seine einzelnen ASCII-Codewerte zu zerlegen, brauchen wir eine Methode, welche uns gestattet, einzelne Zeichen des Wortes abzuschneiden beziehungsweise herauszupicken. Ich müßte eigentlich String statt Wort sagen, denn das alles gilt natürlich auch für Zeichen und Symbole.

Es wäre schön, wenn man dieses Abschneiden am Anfang, in der Mitte oder am Ende dieses Wortes beziehungsweise Strings machen könnte.

Basic kann alles drei:
- LEFT$ schneidet vom linken Rand des Strings Zeichen heraus
- RIGHT$ tut dasselbe auf der rechten Seite
- MID$ pickt Teile aus der Mitte des Strings heraus.

Natürlich bieten diese Befehle auch die Möglichkeit, anzugeben, wieviele Zeichen abgetrennt werden sollen.

Bei LEFT$ und RIGHT$ ist das nur eine einzige Zahlenangabe. Bei MID$ brauchen wir aber zwei Zahlen. Die eine gibt an, ab wo herausgepickt werden soll, die zweite sagt wieviel.

Im Beispiel wird das schnell klar:

```
→ 10 A$= "MOTORHAUBENVERSCHLUSS"
  20 B$ = LEFT$ (A$,5)
  30 PRINT B$
  40 C$ = RIGHT$ (A$,7)
  50 PRINT C$
  60 D$ = MID$ (A$,6,5)
  70 PRINT D$
```

Zeile 10 legt uns die Basis mit der Zuweisung des langen Wortes an die String-Variable A$.

Jetzt geht's ans Beschneiden. Mit Zeile 20 fangen wir an der linken Seite des Wortes an. Hinter dem Befehl LEFT$ steht in der Klammer zuerst der String, der zerpflückt werden soll, nämlich A$. Die zweite Angabe – nach dem Komma – gibt an, wieviele Zeichen abgeschnitten werden sollen. Im Beispiel sind es 5. Diese 5 ergeben gerade den neuen String »MOTOR«, der in Zeile 30 ausgedruckt wird. Die Schreibweise und Funktion von RIGHT$ ist identisch, halt nur auf der rechten Seite des Wortes A$ wirkend. In Zeile 40 erhält dieser Teil-String den Variablennamen C$. Zeile 50 druckt also die rechten 7 Zeichen des Strings A$ aus, was das Wort »SCHLUSS« ergibt.

Interessant wird es in Zeile 60 beim Befehl MID$. Nach der Angabe des betroffenen Strings A$ steht zuerst die Nummer des Zeichens, ab dem von links gezählt herausgeschnippelt werden soll. Die zweite Zahl daneben gibt an, wieviele Zeichen es sein sollen.

In unserem Beispiel ist das sechste Zeichen von links das »H«, ab da 5 Zeichen weiter – inklusive des »H« – ergeben das Wort »HAUBE«, das in Zeile 70 ausgedruckt wird.

Basic-Befehle Nr. 16 LEFT$(X$,A)
- schneidet vom String X$ von links her A-Zeichen ab und bildet daraus einen neuen String.

Nr. 17 RIGHT$(X$,A)
- macht genau dasselbe wie Nr. 16, aber von rechts her.

Nr. 18 MID$(X$,B,A)
- schneidet vom String X$ von links her ab dem B-Zeichen insgesamt A-Zeichen heraus
- bei MID$ kann die zweite Zahl »A« weggelassen werden. Dann schneidet es ab dem B-Zeichen den Rest des Strings ab

Ich empfehle Ihnen, ein bißchen zu experimentieren, am besten mit nur einer Zeile

```
→ 100 PRINT LEFT$("DRACHEN",2)
  RUN 100
```

Erstaunt Sie die Schreibweise? Das sollte jetzt eigentlich nicht mehr passieren.

Der einzige Unterschied zu vorher ist, daß wir vorher den String zuerst einer String-Variablen zugeordnet haben, wodurch er im Speicher aufbewahrt wird. Hier schreiben wir den String direkt an die Stelle der Variablen, der Computer merkt ihn sich halt nicht. Aber mit RUN 100 können Sie leicht den Befehl immer wieder ausführen, nachdem Sie mit dem Cursor auf die Zahl gefahren sind und diese verändert haben.

Genau so geht es mit dem Befehl RIGHT$, weswegen ich ihn hier auslasse.

MID$ ist der schwierigste der drei Befehle.

```
→ 200 PRINT MID$("DRACHEN",2,5)
```





ergibt »RACHE«. Wenn Sie die 2. Zahl ganz weglassen, erhalten Sie »RACHEN«.

Die Frage stellt sich, ob wir aus dem Wort »DRACHEN« mit diesem Befehl zwei getrennte Teile herausholen und so zusammensetzen können, daß zum Beispiel das Wort »RAHE« entsteht.

Das geht!

Wir müssen bloß zuerst den Teil »RA« und in einem zweiten Schritt den Teil »HE« herauszwicken und dann beide Teil-Strings zu einem neuen String zusammenfügen. Die folgenden Zeilen schaffen das:

```
→ 10 A$="DRACHEN"
  20 X$=MID$(A$,2,2)
  30 Y$=MID$(A$,5,2)
  40 Z$=X$+Y$
  50 PRINT Z$
```

Ich glaube, das brauche ich nicht zeilenweise zu erläutern. Jetzt soll aber noch ein bißchen Mathematik dazukommen.

Es ist ja nicht zwingend vorgeschrieben, daß die Zahlen in der Klammer hinter dem String-Befehl konstant sind. Schauen Sie sich das folgende Beispiel an:

```
→ 10 A$="DRACHEN"
  20 X=X+1
  30 B$=LEFT$(A$,X)
  40 PRINT B$
  50 IF X=0 THEN END
  60 GOTO 20
```

Dieses Programm druckt uns das folgende Muster aus:

D
DR
DRA
DRAC
DRACH
DRACHE
DRACHEN

Das Geheimnis liegt in Zeile 30, in der die Zahl der abzuschneidenden Zeichen nicht konstant, sondern durch die Zählschleife mit X von 1 bis 7 hochgezählt wird. Und jetzt kommen wir unserem früheren Wunsch, alle Buchstaben eines Wortes in ihren ASCII-Code umzuwandeln, schon näher. Erinnern Sie sich, diese Umwandlung liefert uns der ASC-Befehl, aber immer nur für den ersten Buchstaben eines Wortes.

Was wir brauchen, ist so ein Muster:

DRACHEN
RACHEN
ACHEN
CHEN
HEN
EN
N

Mit LEFT$ geht das nicht, denn wir schneiden ja an der rechten Seite ab. Außerdem fangen wir mit der vollen Wortlänge von 7 an und reduzieren sie laufend.

Also, der RIGHT$-Befehl muß her, kombiniert mit einer Zählschleife, die aber rückwärts zählt:

```
→ 10 A$="DRACHEN"
  20 X=7
  30 B$=RIGHT$(A$,X)
  40 PRINT B$
  50 X=X-1
  60 IF X=0 THEN END
  70 GOTO 30
```

Die rückwärts zählende Schleife wird durch die Festlegung des Anfangswertes in Zeile 20 und durch das laufende Vermindern der Zählvariablen in Zeile 50 gebildet.

Um noch den ASCII-Codewert der jeweils ersten Buchstaben zu erhalten, müssen wir nur Zeile 40 erweitern zu:

```
→ 40 PRINT B$ "=" ASC(B$)
```

Nach dem bisherigen Ausdruck des Teil-Strings B$ folgt das Gleichheitszeichen – als String zwischen Gänsefüße gesetzt – und danach der umgewandelte Wert des ersten Buchstabens von B$.

Ich muß Ihnen gestehen, daß das alles nur zur Übung war. Denn am elegantesten geht es mit dem Befehl MID$.

Wir kehren wieder zur hochzählenden Schleife zurück und zwicken von links her der Reihe nach die Buchstaben heraus, immer nur einen, mit dem Befehl MID$(A$,X,1)

```
→ 10 A$="DRACHEN"
  20 X=X+1
  30 B$=MID$(A$,X,1)
  40 PRINT B$ "=" ASC(B$)
  50 IF X=7 THEN END
  60 GOTO 20
```

Es gibt noch einen String-Befehl, der uns das Zählen der Zeichenzahl für die Prüfung in Zeile 50 abnimmt. Er heißt LEN(String)

Er ist eine Abkürzung von »LENGTH«, was auf deutsch »Länge« heißt. Er wird seinem Namen gerecht und bestimmt die Länge des Strings. Wenn Sie direkt eingeben:

```
→ PRINT LEN("DRACHEN")
```

dann erhalten Sie die Zahl 7 als Resultat.

Um ihn im obigen letzten Programmbeispiel einzusetzen, brauchen wir lediglich die Zeile 50 ändern:

```
→ 50 IF X=LEN(A$) THEN END
```

So prüft sie X solange, bis es den Wert von LEN(A$), nämlich 7 erreicht hat.

Basic-Befehl Nr. 19 LEN(String)

- gibt die gesamte Länge des Strings einschließlich der Leerzeichen und Steuerzeichen als Zahl aus.
- In der Klammer kann ein String oder aber eine String-Variable sein, der ein String zugewiesen worden ist.

Ich möchte gern einen Rücksprung machen zu der Tabelle der Steuertaste und ihrer ASCII-Codewerte.

Da sehen Sie als untersten Block die Funktionstasten f1 bis f8 angegeben. In diesem Kurs werden sie da praktisch zum ersten Mal erwähnt. Jetzt wird es langsam Zeit, näher darauf einzugehen.

Die Funktionstasten

Wenn Sie eine der Funktionstasten drücken, passiert bekanntlich gar nichts – oder doch?

Nun, sie haben einen eigenen ASCII-Code. Mit einem kleinen Programm, das wir bei der Besprechung der ASCII-Codes verwendet haben, können wir diese Behauptung hier noch einmal überprüfen. Das Programm sah so aus:

```
→ 10 INPUT A$
  20 A=ASC(A$)
  30 PRINT A
  40 GOTO 10
```

Nicht nur die Funktionstasten, sondern auch alle anderen Steuertasten geben so ihren ASCII-Code preis. Den so erhaltenen Code können wir zur Abfrage der Tasten verwenden. Um zu sehen, ob sie gedrückt worden sind. Das geht so: Fügen Sie bitte die folgende Zeile 35 ein:

```
→ 35 IF A=65 THEN PRINT "SERVUS"
```

Das Programm funktioniert wie vorher, nur wenn wir das »A« drücken, dessen ASCII-Code 65 ist, wird der freundliche Gruß ausgedruckt. Statt 65 können Sie jeden beliebigen Codewert verwenden und damit das Drücken der entsprechenden Taste abfragen. Halt! Das stimmt nicht ganz. Denn wenn Sie in Zeile 35 den Codewert einer der Funktionstasten eingeben, rührt sich kein SERVUS.

Warum nicht?

Vielleicht erinnern Sie sich, daß im Unterschied zum GET-Befehl, der im Tastaturpuffer nachschaut, ob dort ein Zeichen abgespeichert ist, der INPUT-Befehl alle eingegebenen Zeichen erst auf den Bildschirm bringt und sie von dort dann weiterverarbeitet.

Das ist des Rätsels Lösung, denn die Funktionstasten hinterlassen halt keine Spur auf dem Bildschirm, genausowenig wie alle anderen Steuertasten.

Das heißt, zu deren Abfrage müssen wir den GET-Befehl nehmen. Ändern wir also die Zeile 10 ab:

```
→ 10 GET A$:IF A$="" THEN 10
```

Und jetzt geht es !

Zur Demonstration der Abfrage der Funktionstasten habe ich unser kleines Programm wie folgt erweitert:

```
→ 10 GET A$:IF A$="" THEN 10
  20 A=ASC(A$)
  30 PRINT A
  35 IF A=133 THEN PRINT CHR$(158)
  36 IF A=134 THEN PRINT CHR$(154)
  37 IF A=135 THEN PRINT CHR$(19)
  38 IF A=136 THEN PRINT CHR$(147)
  40 GOTO 10
```

Die Zeilen 35 bis 38 bilden einen Abfrageblock für die 4 Funktionstasten f1, f3, f5 und f7.

Wenn ihr ASCII-Code auftritt, dann drucken sie das Zeichen des hinter dem jeweiligen CHR$-Befehl stehenden Codewertes aus. Da diese Codewerte aber die Werte der Steuerzeichen – der Reihe nach – GELB, HELLBLAU, CURSOR HOME und BILDSCHIRM LÖSCHEN sind, werden diese Funktionen ausgeführt.

Das ist also das Kochrezept zur Nutzbarmachung der Funktionstasten. Aber auch alle anderen Tasten können so abgefragt und zur Steuerung irgendwelcher Programmschritte verwendet werden. Wir werden diese Methode auch noch benützen.

Übrigens, wenn Sie nur eine einzige Taste abfragen, deren Codewert aber nicht ausdrucken beziehungsweise nicht weiterverwenden, geht die Abfrage von oben dadurch schneller, daß Sie den String gar nicht erst lange umwandeln:

```
→ 10 GET A$: IF A$="" THEN 10
  35 IF A$=CHR$(133) THEN PRINT CHR$(150)
  40  und so weiter
```

Die Prüfzeile 35 vergleicht einfach den eingegebenen String A$ mit dem String, der dem ASCII-Code 133 entspricht – der CHR$-Befehl macht das möglich. Nur bei vielen Abfrage-IFs ist das Tippen der vielen CHR$ mühsam und die ASC-Methode von vorher hat ihre Berechtigung.

Ein Programm mit den String-Befehlen

Ich möchte mit Ihnen zusammen ein drittes Programm entwickeln, welches hauptsächlich die String-Befehle verwendet und Ihnen ein bißchen Erfahrung in der String-Verarbeitung gibt.

Aufgabe:

Im ersten Teil des Programms sollen eingegebene Wörter in einen Geheimcode umgewandelt werden. Die verwandelten Wörter stehen alle auf dem Bildschirm. Per Tastendruck soll dann der zweite Programmteil angesprungen werden, in dem die einzelnen Codewörter wieder in ihre ursprüngliche Form zurückgewandelt (dekodiert) werden.

Der übersichtlichen schrittweisen Darstellung zuliebe verzichte ich auf ein Flußdiagramm. Wir »hacken« uns also durch. Das komplette Programm finden Sie auf Seite 63. Listing 3 zeigt, wie man in Basic »KODIEREN/DEKODIEREN« kann.

Die Eingabe der Wörter machen wir zuerst über den GET-Befehl. Sie wissen, daß der GET-Befehl immer nur ein Zeichen annimmt, aber keine ganzen Wörter. Wir müssen also erst eine Schleife bauen, um die einzelnen Zeichen zu einem Wort zusammenzusetzen.

```
→ 100 GET W$: IF W$="" THEN 100
  120 PRINT W$;
  130 A$=A$+W$
  150 GOTO 100
```

Zeile 100 ist die übliche Eingabe-Warteschleife mit GET.

Zeile 120 druckt jedes einzelne Zeichen W$ mit dem Semikolon aneinandergeklebt aus. Zeile 130 ist etwas Neues. Es wird ein String mit Namen A$ dadurch gebildet, daß zu seinem jeweiligen Zeichen – am Anfang ist keines da – das gerade eingegebene Zeichen W$ dazugenommen wird. Auf diese Weise entsteht ein ganzes Wort A$. Der Rücksprung in Zeile 150 läßt dieses Wort unendlich lange werden.

Wir brauchen also am Ende des Wortes einen Aussprung aus der Schleife. Ich schlage vor, das Wort mit der RETURN-Taste zu beenden. Deshalb müssen wir in Zeile 140 den ASCII-Code der RETURN-Taste abfragen. Er ist 13.

```
→ 140 IF W$=CHR$(13) THEN 170
  170 PRINT"**"
```

Die Abfrage kennen wir schon. Sie springt nach 170, wo vorläufig zum Ausprobieren zwei Sterne ausgedruckt werden.

Kodier/Dekodier-Algorithmus

»Algorithmus« ist die Bezeichnung für einen logischen Vorgang oder eine Formel, in unserem Fall die Vorschrift, nach der das Wort A$ verschlüsselt werden soll.

Ich habe einen sehr leichten Algorithmus gewählt:

Der ASCII-Codewert jedes Zeichens wird um die Zahl seines Platzes im Wort erhöht, der erste Buchstabe um 1, der 2. Buchstabe um 2 und so weiter.

Das Dekodieren geht entsprechend, indem immer die Platz-Ziffer abgezogen wird.

Also müssen wir zuerst das Wort A$ in seine ASCII-Codewerte umwandeln. Das haben wir schon einmal gemacht.

In Zeile 160 gebe ich ein Wort für A$ vor, aber nur zum Ausprobieren.

```
→ 160 A$="WERT"
  170 X=X+1
  180 A=ASC(MID$(A$,X,1))
  190 B=A+X
  195 PRINT B
  220 IF X <> LEN(A$) THEN 170
```

A$ in Zeile 160 sei also das fertig eingegebene Wort.

Die Zeilen 170 und 220 bilden eine Zählschleife, deren Variable X von 1 bis zur Länge des eingegebenen Wortes A$ zählt und am Ende weiterspringt. In unserem Beispiel ist LEN(A$)=4 Buchstaben.

Zeile 180 kennen Sie schon. Sie wandelt jeden Buchstaben des Wortes A$ – einzeln wegen der letzten Ziffer 1 – von links ab dem X. Zeichen in seinen ASCII-Code um.

Zeile 190 ist die eigentliche Verschlüsselung oder Kodierung. Hier wird eine Variable B dadurch gebildet, daß zum ASCII-Wert »A« des gerade umgewandelten Zeichens »A$« der jeweilige Wert von X dazugezählt wird.

Zeile 195 ist eingeschoben, um die neuen ASCII-Werte auszudrucken.

Starten Sie diesen Teil mit RUN 160 und Sie erhalten die neue Zahlenfolge 88, 71, 85 und 88.

Jetzt nehmen wir Zeile 160 heraus, oder beser noch, wir wandeln sie in eine REM-Zeile zur Schönschrift um.

Sie können jetzt mit RUN das Ganze laufen lassen, nämlich Eingabe eines Wortes bis zum Ausdruck der neuen Codezahlen.

Funktion	Taste(n)	Zeichen	ASCII
schwarz	CTRL-1		144
weiß	CTRL-2		5
rot	CTRL-3		28
lila	CTRL-4		159
purpur	CTRL-5		156
grün	CTRL-6		30
blau	CTRL-7		31
gelb	CTRL-7		158
orange	C=-1		129
braun	C=-2		149
hellrot	C=-3		150
hellgrau	C=-4		151
mittelgrau	C=-5		152
hellgrün	C=-6		153
hellblau	C=-7		154
dunkelgrau	C=-8		155
Revers ein	CTRL-9		18
Revers aus	CTRL-0		146
Cursor rauf	SHIFT-CRSR↑		145
Cursor ab	CRSR↓		17
Cursor links	SHIFT-CRSR←		157
Cursor rechts	CRSR→		29
Schirm löschen	SHIFT-CLR/HOME		147
Cursor Home	CLR/HOME		19
Insert	SHIFT-INST/DEL		148
Delete	INST/DEL		20
Klein/Groß-buchstaben	SHIFT-C=		14
Großbuch-staben/Zeichen	SHIFT-C=		142
Funktionstaste f1	f1		133
f2	SHIFT-f1		137
f3	f3		134
f4	SHIFT-f3		138
f5	f5		135
f6	SHIFT-f6		139
f7	f7		136
f8	SHIFT-f7		140

Tabelle 1. Die Steuertasten und ihre reversen Zeichen

Jetzt allerdings erhalten Sie immer eine Zahl mehr, als Zeichen eingegeben wurden. Das ist die RETURN-Taste.

Die nächsten drei Zeilen wandeln die verschlüsselten Codezahlen »B« in Buchstaben »B$« zurück und setzen sie zu einem verschlüsselten Wort »C$« zusammen.

```
→ 200 B$=CHR$(B)
  210 C$=C$+B$
  240 PRINT C$
```

Mit RUN erhalten Sie jetzt auch das verschlüsselte Wort ausgedruckt.

Um mehrere Wörter eingeben zu können, bilden wir mit dem GOTO-Befehl der Zeile 260 eine Schleife. Zeile 195 kann jetzt entfernt werden.

```
→ 195
  260 GOTO 100
```

Die Eingabe eines zweiten Wortes stößt auf Schwierigkeiten. Der Grund dafür liegt darin, daß das erste Wort A$ noch vorhanden ist. Wir müssen es einfach löschen. Und weil wir gerade beim Korrigieren sind: Auch die Zählvariable X muß vor der nächsten Eingabe auf 0 gesetzt werden. Der letzte Fehler besteht darin, daß in Zeile 220 der LEN-Befehl immer ein Zeichen mehr mißt, weil ja die RETURN-Taste am Ende des eingegebenen Wortes für ihn auch als Zeichen gewertet wird.

Hier bietet sich zusätzlich die Gelegenheit, für die Schönschrift der Ausgabe etwas zu tun. Mir gefällt nämlich nicht, daß beim Ausdrucken zwischen den kodierten Wörtern kein Zwischenraum gelassen wird. Ich schiebe daher die Zeile 230 vor den PRINT-Befehl, um an den bestehenden String »C$« – das kodierte Wort – ein Leerzeichen mit dem ASCII-Code 32 anzuhängen.

Die folgenden Zeilen korrigieren das alles.

```
→ 220 IF X <> LEN(A$)-1 THEN 170
  230 C$=C$+CHR$(32)
  250 A$=""
  260 X=0: GOTO 100
```

In Zeile 220 prüft der IF-THEN Befehl auf ein Zeichen weniger, als das Wort plus RETURN-Taste lang ist. Zeile 250 zeigt, wie man eine String-Variable durch Zuweisung eines Null-Strings löscht. Dasselbe tut Zeile 260 mit der numerischen Variablen X.

Jetzt läuft der erste Programmteil, der aus einem eingegebenen Wort ein verschlüsseltes Wort baut. Wir erhalten zum Beispiel für das Wort »WERT« den Code »XGUX«.

Das Dekodieren stelle ich mir so vor, daß der Benutzer per Tastatur die Codes eingibt, welche er entschlüsseln will. Das wollen wir zur Übung diesmal mit INPUT machen.

```
→ 280 INPUT E$
```

Die nächsten Zeilen gleichen weitgehend den Zeilen 170 bis 260, da der Vorgang ja fast identisch ist. Eine Ausnahme besteht zur Zeile 190, da wir zum Dekodieren ja die Zählvariable X abziehen müssen. Außerdem habe ich die beiden Zeilen 200 und 210 zu einem Ausdruck zusammengefaßt. Und schließlich braucht dem LEN-Befehl keine 1 abgezogen werden, da beim INPUT-Befehl im Gegensatz zum GET die eingegebene RETURN-Taste nicht zum Wort dazugerechnet wird.

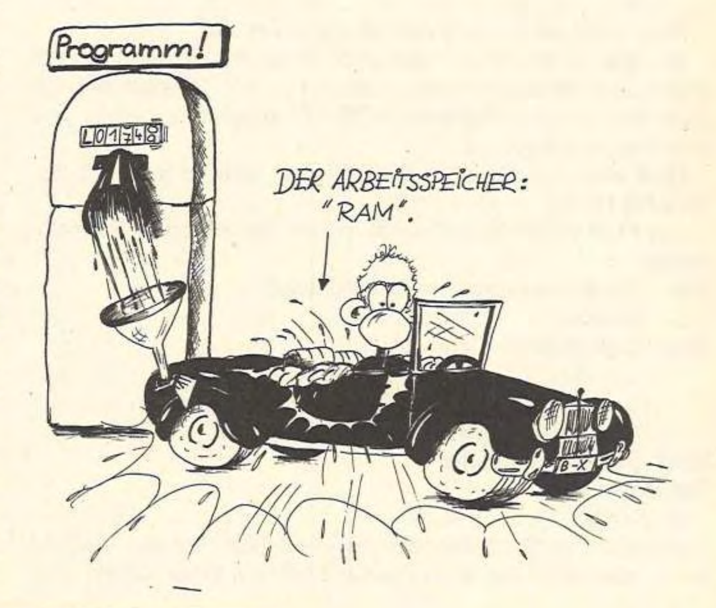
Also sieht das Dekodieren so aus:

```
→ 280 INPUT E$
  290 X=X+1
  300 E=ASC(MID$(E$,X,1))
  310 D=E-X
  320 D$=D$+CHR$(D)
  330 IF X <> LEN(E$) THEN 290
  340 D$=D$+CHR$(32)
  350 PRINT D$
  360 E$=""
  370 X=0:GOTO 280
```

Mit RUN 280 können Sie diesen Programmteil separat testen.

Jetzt müssen wir die beiden Teile noch zusammenbinden. Ich stelle mir vor, daß es praktisch ist, nach einigen eingegebenen Wörtern diese zu dekodieren. Das Signal dazu soll die Funktionstaste F1 sein. Wenn sie also gedrückt wird, soll das Programm vom Kodierteil auf den Dekodierteil springen.

Diese Prüfung müssen wir ganz am Anfang des Kodierteils vornehmen, und die GET-Eingabeschleife bietet sich dazu in



hervorragender Weise an. Wir schieben also die folgenden drei Zeilen ein:

```
→ 110 IF W$=CHR$(133) THEN 270
  270 PRINT"DEKODIEREN"
  285 IF E$="ENDE" THEN END
```

Zeile 110 verwendet den ASCII-Code von 133 der f1-Taste, um auf die Zeile vor dem INPUT-Befehl zu springen (270), wo ein Hinweis ausgedruckt wird.

Die Zeile 285 beendet auf simple Weise das Programm. Das einzige, was nicht passieren darf ist, daß das Wort »ENDE« als kodierter Wert auftritt.

Zur Bedienung des Programms ist zu sagen, daß Sie beim Dekodieren lediglich die auf dem Bildschirm stehenden verschlüsselten Wörter einzeln ablesen und eintippen müssen.

Wem das zu mühsam ist, soll ein Programm für die automatische Dekodierung schreiben.

FAZIT

1. Eine String-Variable kann dadurch »gelöscht« werden, daß man ihr einen Null-String zuweist mit dem Befehl A$=" ". Das entspricht dem Nullsetzen einer numerischen Variablen mit X=0.
2. Funktionstasten und andere Steuertasten, die nicht auf dem Bildschirm erscheinen, können nur mit dem GET-Befehl abgefragt werden.
3. Ein String kann durch Hinzufügen von anderen Strings zu einem größeren String gleichen Namens erweitert werden: A$=A$+B$

Schleifen mit eigenem Befehl

Programmschleifen sind uns nichts Neues mehr. In jedem Programm sind sie in irgendeiner Form vorgekommen. Wir haben sie bis jetzt – wie ich zugeben muß – in sehr »altmodischer« Art programmiert, nämlich mit Hochzählen einer Schleifenvariablen und Prüfen, ob die Aussprungsbedingung schon erreicht ist.

Basic stellt uns dafür einen sehr bequemen Befehl zur Verfügung. Er besteht aus 3 Wörtern:
FOR..TO..NEXT

Auf deutsch läßt sich das übersetzen mit »Für..Bis..Der Nächste«.

Ein Beispiel soll ihn erklären. Links steht die alte Zählschleife, rechts die neue Befehlsfolge.

```
→ 10 X=X+1              10 FOR X=1 TO 10
  20 PRINT X            20 PRINT X
  30 IF X=10 THEN END
  40 GOTO 10            40 NEXT X
```

Man sieht sehr schön den Zusammenhang.

Die alte Methode der Definition einer immer um 1 höherzählenden Variablen mitsamt einer Prüfung ist jetzt in einer Zeile mit den Befehlsteilen FOR TO zusammengefaßt. Die Prüfung ist eingebaut.

Dem alten Rücksprung mit GOTO entspricht jetzt der Befehlsteil NEXT.

Die FOR-NEXT-Schleife hat immer die folgende Schreibweise:

FOR Schleifenvariable = Anfangswert
TO Endwert
STEP Schrittweite
.
.
(Programm innerhalb der Schleife)
NEXT Schleifenvariable

Die Schleife

```
FOR T=0 TO 14 STEP 2
```

setzt also die Schleifenvariable T auf Null und zählt bis 14 hoch, allerdings immer in Zweierschritten. Wenn STEP weggelassen wird, dann wird automatisch die Schrittweite 1 genommen

Eine rückwärtszählende Schleife sieht so aus:

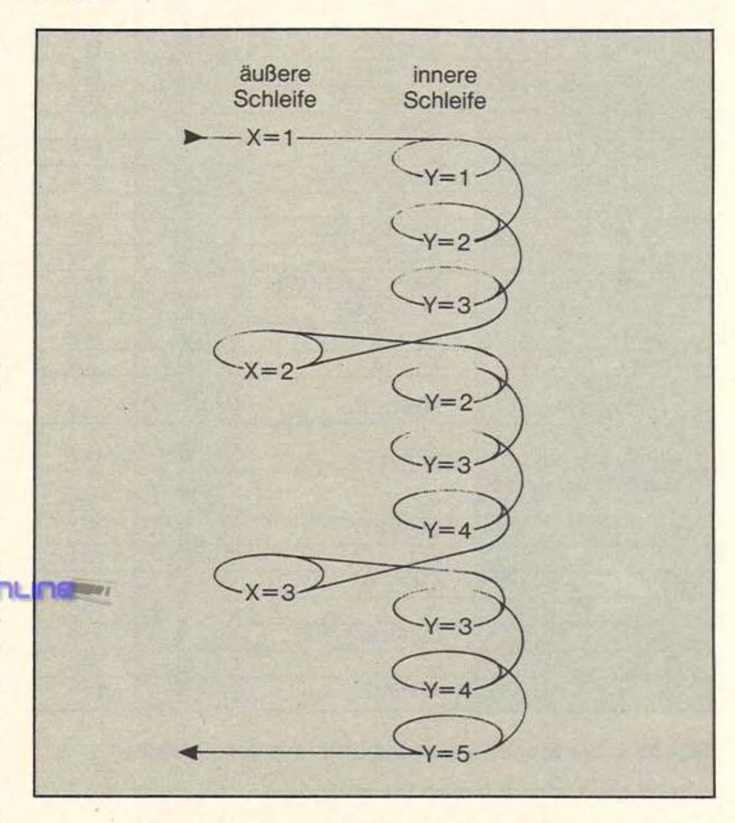
```
FOR T=14 TO 0 STEP -1
```

In diesem Fall muß die Schrittweite, weil sie negativ ist, immer angegeben werden.

Interessant ist auch, daß die Schrittweite ein Dezimalbruch sein kann.

```
→ 10 FOR X=1 TO 15 STEP 3.7
  20 PRINT X
  30 NEXT X
```

ergibt die Zahlen 1, 4.7, 8.4, 12.1 – also jeweils um 3.7 erhöht.



Daraus ist ersichtlich, daß der NEXT-Befehl in der Schleife nicht prüft, ob die Schleifenvariable X den Endwert 15 exakt erreicht hat, sondern ob sie größer ist.

FOR-TO-NEXT erlauben uns auch, Schleifen zu »schachteln«. Was das bedeutet, zeigt uns das nächste Beispiel:

```
10 FOR X=1 TO 3
20    FOR Y=X TO X+2
30   PRINT Y
40    NEXT Y
50 NEXT X
```

Zeile 10 und 50 bilden die äußere Schleife. Die Zeilen 20 bis 40, die ich zur besseren Übersicht weiter eingerückt habe, bilden die innere Schleife.

Wir wollen nachvollziehen, was da abläuft.

Sie sehen, daß die innere Schleife so oft wiederholt wird, wie es die äußere Schleife vorgibt.

Der NEXT-Befehl hat noch zwei Feinheiten. Bei einer einzigen Schleife kann die Angabe der Schleifenvariablen hinter ihm wegfallen. Die folgende kleine Zeitverzögerungsschleife demonstriert das:

```
→ FOR T=1 TO 500: NEXT
```

Bei geschachtelten Schleifen stehen oft mehrere abschließende NEXT-Befehle hintereinander. Statt:

```
40 NEXT Y
50 NEXT X
```

kann geschrieben werden:

```
40 NEXT Y,X
```

Dabei ist auf die Reihenfolge der Schleifenvariablen zu achten. Zuerst kommt immer die innere Variable.

Neben allen diesen geschilderten Feinheiten hat die FOR-TO-NEXT-Schleife noch einen weiteren Vorteil gegenüber der Zählschleife mit IF-THEN: Sie ist wesentlich schneller in der Ausführungszeit. Wenn Sie dieses Phänomen interessiert, dann lesen Sie bitte den Beitrag »SO MACHT MAN PROGRAMME SCHNELLER« im 64er Sonderheft 2/86 ab Seite 44.

Basic-Befehl Nr. 20 FOR-TO STEP-NEXT

- hinter FOR steht die Schleifenvariable mit ihrem zugewiesenen Anfangswert. Hinter TO steht der Endwert der Schleifenvariable.
- mit STEP kann die Schrittweite eingestellt werden. Sie kann auch negativ oder ein Dezimalbruch sein.
- rückwärts zählende Schleifen müssen immer eine negative Schrittweite mit STEP angeben.
- mehrere Schleifen können geschachtelt werden. Die innere Schleife muß vor der äußeren abgearbeitet werden.
- bei NEXT kann die Schleifenvariable weggelassen werden, wenn dadurch keine Zweideutigkeiten mit anderen Schleifenvariablen entstehen.
- wenn mehrere NEXT-Befehle hintereinander auftreten, können ihre Schleifenvariablen durch Kommata getrennt hinter einem einzigen NEXT stehen. Die Reihenfolge »innere Variable vor äußerer Variable« ist einzuhalten.

Als anwendbares Beispiel für die FOR-NEXT-Schleife greife ich im folgenden Programm den zweiten Teil des KODIER/DEKODIER-Programms heraus und versehe ihn aber mit den neuen Schleifenbefehlen. Den ersten Teil können Sie sich selbst umschreiben.

```
→ 270 PRINT"DEKODIEREN"
  280 INPUT E$
  285 IF E$="ENDE" THEN END
  290 FOR X=1 TO LEN(E$)
  300 E=ASC(MID$(E$,X,1))
  310 D=E-X
  320 D$=D$+CHR$(D)
  330 NEXT
  340 D$=D$+CHR$(32)
  350 PRINT D$
  360 E$=""
  370 GOTO 280
```

Der Unterschied liegt in den Zeilen 290, 330 und 370.

Daten aus dem Keller holen

Es gibt Situationen im Programmierleben, in denen es wünschenswert wäre, auf einen ganzen Stall voll Zahlen oder Wörter zurückgreifen zu können, ohne sie jedesmal per INPUT oder GET mühsam in den Computer eingeben zu müssen. Einmal eingegeben, sollten Sie immer zur Verfügung stehen.

Diese Anforderung zeigt deutlich in Richtung »Speicherung von Daten« im Computer. Nun, wenn wir einer String-Variablen in einer Befehlszeile einen String zuordnen, wird dieser bekanntlich auch gespeichert. Aber das würde dazu führen, daß wir beispielsweise fünf Zuordnungen machen müßten, um fünf Strings zu speichern, etwa so:

```
→ 10 A$="BUCH"
  20 B$="SCHULTER"
  30 C$="SCHIFF"
  40 D$="TISCH"
  50 E$="MALEREI"
```

Bei fünf Strings geht es ja noch, aber bei 30 oder gar bei 200?

Diesen Notstand stellt Basic ab mit dem Befehl

DATA

Hinter diesem Befehl können in einer Befehlszeile soviel Zahlenwerte oder Strings geschrieben und dadurch gespeichert werden, wie in einer Programmzeile Platz haben.

Alle Eintragungen in einer DATA-Zeile müssen durch Kommata getrennt sein.

Das Beispiel oben sieht jetzt so aus:

```
→ 10 DATA BUCH,SCHULTER,SCHIFF,TISCH,MALEREI
```

Sie sehen, die Strings brauchen nicht in Gänsefüßen stehen, mit einer Ausnahme. Diese ist weiter unten im Kasten beschrieben.

Eine DATA-Zeile mit Zahlen sieht so aus:

```
→ 20 DATA 25,123,225,16,24
```

Jetzt muß noch geklärt werden, wie man eigentlich die gespeicherten Daten herausholen kann.

Nun, das geht erstens immer der Reihe nach und zweitens mit dem Basic-Befehl

READ

was leicht und treffend mit »LESEN« übersetzbar ist. Probieren wir es:

```
→ 40 READ A$
  50 PRINT A$
```

Nach RUN druckt das Programm das erste der Wörter in der DATA-Zeile, nämlich »BUCH« aus. An die anderen kommen wir nicht dran.

Wenn Sie in Zeile 40 und 50 die Variable in eine numerische Variable verwandeln, kommen wir trotzdem nicht an die Zahlen. Zusätzlich bestraft uns der Computer mit einer Fehlermeldung.

Bauen wir also eine Schleife ein, um den READ-Befehl 5 x zu wiederholen.

```
→ 30 FOR X=1 TO 5
  60 NEXT
```

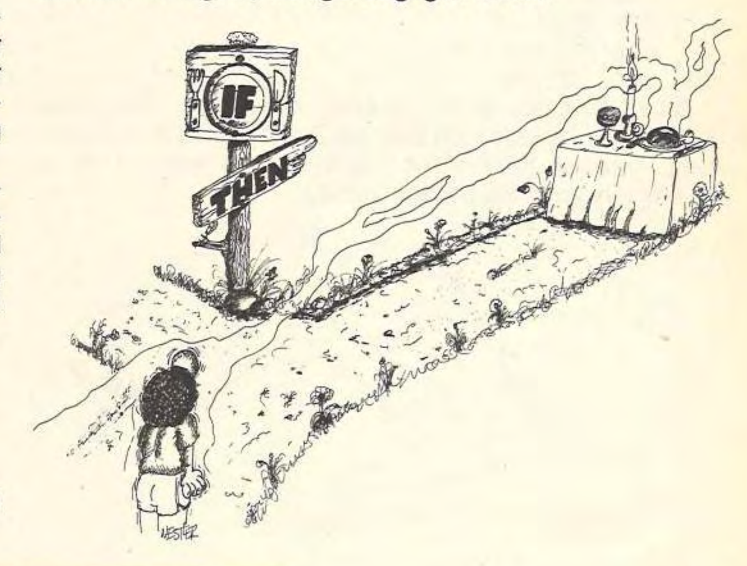

Jetzt tauchen alle fünf Wörter auf. Eine Erhöhung der Schleifenvariablen X auf 6 quittiert der Computer nach RUN wieder mit einer Fehlermeldung, da wir mit der Stringvariablen A$ in den Bereich der Zahlen eingedrungen sind, wodurch Variablentyp und Wert nicht zusammenpassen.

Eine zweite Schleife schafft Abhilfe:

```
→ 70 FOR X=1 TO 5
  80 READ A
  90 PRINT A
  100 NEXT
```

Wenigstens soweit haben wir es gebracht, daß wir alle gespeicherten Daten aus dem Speicher holen können.

Erproben Sie noch durch eine Erhöhung des Endwertes der Schleife in Zeile 70 auf 10, was passiert, wenn wir mehr Daten lesen wollen, als vorhanden sind. Sie werden merken, daß das nicht geht, weniger dagegen schon.



Wir wollen uns jetzt auf nur einen Datentyp beschränken. Löschen Sie bitte die Zeilen 70 bis 100.

Ein weiteres Experiment besteht darin, den READ-Vorgang endlos zu wiederholen mit einer neuen Zeile 70.

```
→ 10 DATA BUCH,SCHULTER,SCHIFF,TISCH,MALEREI
  30 FOR X=1 TO 5
  40 READ A$
  50 PRINT A$
  60 NEXT
  70 GOTO 30
```

Auch dieser Versuch schlägt fehl. Der Grund dafür liegt im Verfahren des READ-Befehls, nämlich einen internen Zähler mitlaufen zu lassen, der anzeigt, auf das wievielte Wort der READ-Befehl zugreifen muß. Und im obigen Fall ist dieser Zähler durch die Wiederholung mit GOTO 30 über die vorhandenen fünf Wörter hinausgelaufen.

Ein zu READ-DATA gehörender Basic-Befehl stellt diesen Zähler wieder an seinen Anfang zurück. Der Befehl lautet: RESTORE
was soviel heißt wie »wiederherstellen«. Oben vor den GOTO 30 Befehl gestellt setzt RESTORE in der Tat alles zurecht:

```
→ 65 RESTORE
```

Die jetzt erreichte Endlos-Schleife kann mit der STOP-Taste unterbrochen werden.

Basic-Befehle Nr.21, 22 und 23
READ DATA RESTORE

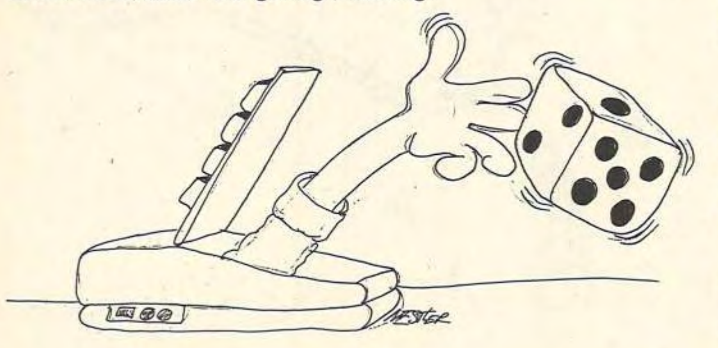
- Mit DATA können beliebig viele Zahlen und Strings in einem Programm gespeichert werden.
- Sie stehen hinter dem DATA-Befehl, durch Kommata voneinander getrennt.
- Strings brauchen nicht zwischen Gänsefüßen stehen, es sei denn, sie enthalten als Teil des Strings ein Komma oder ein Semikolon. Da diese besondere Steuerzwecke erfüllen, muß in diesem Fall der String zwischen Gänsefüße gesetzt werden.
- Mit dem Befehl READ werden die Daten der Reihe nach gelesen. Datentyp und Variablentyp müssen einander entsprechen.
- Sollen die Daten mehrfach ausgelesen werden, muß mit RESTORE der Anfangszustand des Auslesens hergestellt werden.

Ein lustiges Experiment will ich Ihnen noch zeigen, bevor wir ein letztes Programm angehen.

Schreiben Sie die FOR-NEXT Zeile 30 so, wie angegeben und vertauschen Sie NEXT mit PRINT A$.

```
→ 30 FOR X=1 TO INT(RND(0)*5)+1
  40 READ A$
  50 NEXT
  60 PRINT A$
  70 RESTORE
  80 GOTO 30
```

Zeile 30 müßte Ihnen bekannt vorkommen. Die obere Grenze der Schleife wird jetzt per Zufall bestimmt. Sie kann nach unserem Kochrezept von früher die Werte 1 bis 5 annehmen, aber wie gesagt, zufällig.



Da der PRINT-Befehl erst nach der Schleife kommt, druckt er nicht alle Wörter, sondern nur das zuletzt gelesene aus. Wir können somit eine zufällige Auswahl aus der Menge der gespeicherten Wörter treffen, was wir gleich beim letzten Programm verwenden werden.

Ein Programm mit Read-Data

Das ganze Programm ist im Listing 4 »BEGRIFFE RATEN« abgedruckt.

Es stammt ebenfalls aus dem schon mehrfach zitierten Buch »Computerspiele und Knobeleien« von Rüdeger Baumann.

Aufgabe:

- Der Computer hat viele Begriffe gespeichert. In unserem Fall sind es 14 Sportarten.
- Per Zufallsgenerator wird einer der Begriffe ausgewählt. Nur seine Länge wird dem Spieler durch Punkte bekanntgegeben.
- Der Spieler rät einen Buchstaben.
- Das Programm vergleicht diesen Buchstaben mit allen Buchstaben des Begriffes und schreibt den Buchstaben auf die richtigen Plätze, falls es eine Übereinstimmung feststellt.
- Der Vorgang wird so lange wiederholt, bis das Wort erraten ist.
- Zum Abschluß gibt der Spieler das komplette Wort ein.

Ich möchte das Programm blockweise mit Ihnen durchgehen.

Zeile 10 löscht den Bildschirm – diesmal mit dem ASCII-Code!

Die Zeilen 80 bis 180 enthalten mit PRINT-Befehlen die Spielanleitung.

In den DATA-Zeilen 730 bis 760 sind die Sportarten gespeichert. In Zeile 240 bis 260 wird in der vorher schon verwendeten Art und Weise per Zufall eine Sportart »U$« ausgewählt.

Zeile 290 mißt die Länge des Wortes U$.

Die Zeile 300 erfindet ein Hilfswort »H$«, welches am Anfang nur aus Punkten besteht, und zwar aus genau so vielen, wie der Sportbegriff lang ist.

Jetzt folgt der erste Versuch.

Zeile 370 druckt als Prompt das punktierte Hilfswort »H$« aus, um die Länge anzuzeigen.

In Zeile 390 wird per INPUT ein Buchstabe »L$« angefordert und eingegeben. Wird gleich das ganze Wort eingegeben und ist es richtig, springt die Zeile 400 auf den Abschnitt der Entscheidung über Spielwiederholung ab Zeile 600.

Zeile 410 ist die Sicherung gegen unbeabsichtigtes Drücken der RETURN-Taste ohne Eingabe.

Der schwierigste Teil des Programms ist der Vergleich des eingegebenen Buchstabens mit dem vorgegebenen Sportbegriff. Er beginnt ab Zeile 610.

Zeile 480 und 530 bilden eine Zählschleife in der Länge des vorgegebenen Sportbegriffs »U$«. Innerhalb dieser Schleife holt Zeile 490 die einzelnen Buchstaben aus dem Wort »U$« heraus – mit der Methode des MID$-Befehls. Zeile 500 macht dasselbe im Gleichtakt für das (gepunktete) Hilfswort »H$«.

Zeile 500 vergleicht jetzt zuerst den eingegebenen – geratenen – Buchstaben L$ mit jedem einzelnen Buchstaben B$ des Wortes U$. Ist er identisch, dann wird ein neues Hilfswort »N$« aus dem richtig geratenen Buchstaben L$ gebildet. War der Buchstabe falsch, dann wird aus dem Hilfswort H$ ein Punkt in das neue Hilfswort N$ übernommen.

Das wird für alle Stellen der gleichlangen Wörter U$ und H$ gemacht.

Danach wird in Zeile 540 das Hilfswort N$, das jetzt vielleicht schon eine Kombination aus Punkten und Buchstaben



ist, dem alten Hilfswort H$ zugeordnet, welches ja dem Spieler beim nächsten Ratevorgang gezeigt wird (Zeile 370).

Danach wird der Ratevorgang in Zeile 390 und damit der ganze Ablauf wiederholt.

Jedesmal, wenn ein Buchstabe richtig geraten ist, kommt er an die entsprechende Stelle der Hilfswörter N$ und H$.

Erst wenn der Spieler das ganze Wort kennt und es komplett in Zeile 390 eingibt, springt er, wie schon erwähnt, von Zeile 400 nach Zeile 600, wo von 630 bis 670 in üblicher Manier die Frage nach Wiederholung des Spiels oder Ende gestellt wird.

In Zeilen, die durch ein IF-THEN-Kommando angesprungen werden, dient die Zeilennummer als Ziel. Alle anderen Zeilennummern dienen nur der Lesbarkeit und der Schönschrift des Programms.

Zusammenfassung

Ich habe Sie mit den wichtigsten Befehlen von Basic vertraut gemacht, mit vielen kleinen Beispielen und - wie ich hoffe - in verständlicher Sprache.

Die Befehle, die ich ausgewählt habe, geben Ihnen eine gute Basis für weitere Programmierversuche oder zum Lesen und Verstehen von fremden Programmen.

Dem Charakter einer Einführung entsprechend habe ich einige Befehle, die ich zur höheren Programmierung zähle, nicht erwähnt. Es sind dies:
DIM, DEF, FNR, PEEK, POKE, SYS, USR, WAIT, GOSUB-RETURN und ON.

Auch einige mathematische Funktionen und Cursor-Steuerbefehle sind unter den Tisch gefallen, wie:
ABS, AND, COS, NOT, OR, SIN, TAN, POS, TAB und SPC.
Aber gerade diese können Sie jetzt mit Ihrem Wissen leicht aus dem Handbuch von Commodore erlernen.

Die oberen Befehle aber bedürfen noch einer genaueren Erklärung. In den 64'er Ausgaben gibt es immer wieder Basic-Kurse für Fortgeschrittene, die das erklären, was in diesem Kurs nicht behandelt werden konnte.

Mir bleibt nur noch, meinen eingangs gemachten Vorschlag zu wiederholen:

Wenn Sie Fragen haben, bitte schreiben Sie mir. Ich werde sie entweder direkt beantworten oder aber sammeln und in einem neuen Artikel zusammenfassen.

(Dr. Hauck/cg)



Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel {CLR}, muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksumme 64 auf Seite 135.



```
10 PRINT"{CLR}"                                <254>
20 PRINT"****** WELCHE HAND ********           <024>
40 REM------- SPIELANLEITUNG---------          <255>
50 PRINT                                       <152>
60 PRINT"NEHMEN SIE BITTE IN DIE EINE          <086>
70 PRINT"HAND EINE 10-PFENNIG MUENZE,          <014>
80 PRINT"IN DIE ANDERE HAND EINE               <212>
90 PRINT"1-PFENNIG MUENZE.                     <192>
100 PRINT                                      <202>
110 PRINT"ICH WERDE ERRATEN, IN WELCHER        <034>
120 PRINT"HAND SIE WELCHE MUENZE HABEN,        <229>
130 PRINT"WENN SIE MIR FOLGENDE FRAGE          <053>
140 PRINT"BEANTWORTEN:                         <180>
150 PRINT                                      <252>
160 PRINT"MULTIPLIZIEREN SIE DEN WERT          <022>
170 PRINT"DER MUENZE IN IHRER RECHTEN          <080>
180 PRINT"HAND MIT EINER GERADEN ZAHL,         <240>
190 PRINT"DEN WERT DER MUENZE IN IHRER         <194>
200 PRINT"LINKEN HAND MIT EINER                <230>
210 PRINT"UNGERADEN ZAHL.                      <052>
220 PRINT                                      <066>
230 PRINT"IST DIE SUMME DER VON IHNEN          <047>
240 PRINT"ERRECHNETEN ZAHLEN EINE              <148>
250 PRINT"GERADE ODER UNGERADE ZAHL ?          <042>
260 PRINT                                      <108>
270 :                                          <248>
280 REM------------FRAGE UND ANTWORT----       <112>
290 :                                          <012>
300 INPUT"'GERADE' ODER 'UNGERADE'";A$         <022>
310 IF A$="GERADE" THEN 350                    <242>
320 IF A$="UNGERADE" THEN 380                  <051>
330 PRINT:PRINT                                <028>
340 PRINT"FALSCHE EINGABE":GOTO 300            <060>
350 M10$="LINKS"                               <076>
360 M01$="RECHTS"                              <214>
370 GOTO 400                                   <068>
380 M10$="RECHTS"                              <226>
390 M01$="LINKS"                               <124>
400 PRINT:PRINT                                <098>
410 PRINT"DIE 10-PFENNIG SIND ";M10$           <091>
420 PRINT"DER 1-PFENNIG IST ";M01$             <092>
430 END                                        <178>
```

© 64'er

Listing 1. Welche Hand

```
10 PRINT"{CLR}"                                <254>
20 PRINT"******** ZAHLEN RATEN *******"        <060>
30 PRINT                                       <132>
40 :                                           <016>
50 REM+++++++ VORBEREITUNG +++++++++++         <244>
60 :                                           <036>
70 Z=INT(RND(0)*100)+1                         <060>
80 V=0                                         <065>
90 :                                           <066>
100 REM+++++++ RATEN ++++++++++++++++          <082>
110 :                                          <086>
120 PRINT"ICH HABE MIR EINE ZAHL               <144>
130 PRINT"ZWISCHEN 1 UND 100 GEMERKT.          <252>
140 PRINT                                      <242>
150 INPUT"WELCHE ZAHL IST ES ";R               <020>
160 V=V+1                                      <148>
170 :                                          <146>
180 REM+++++++ PRUEFEN +++++++++++++++         <096>
190 :                                          <166>
200 IF R>Z THEN PRINT R;:GOTO 230              <052>
210 IF R<Z THEN PRINT R;:GOTO 240              <126>
220 IF R=Z THEN PRINT R;:GOTO 250              <152>
230 PRINT"IST ZU GROSS":GOTO 140               <195>
240 PRINT"IST ZU KLEIN":GOTO 140               <043>
250 PRINT"IST RICHTIG"                         <067>
260 PRINT"SIE HABEN";V;"VERSUCHE ";            <020>
270 PRINT"BENOETIGT                            <030>
280 :                                          <002>
290 REM+++++ WIEDERHOLUNG ODER ENDE ++++       <153>
300 :                                          <022>
310 PRINT:PRINT:PRINT                          <181>
320 PRINT"NOCH EINMAL (J/N) ?"                 <074>
330 GET A$: IF A$=""THEN 330                   <205>
340 IF A$="J"THEN 10                           <137>
350 IF A$<>"N" THEN 320                        <191>
360 PRINT:PRINT:PRINT"AUF WIEDERSEHEN          <017>
370 END                                        <118>
```

© 64'er

Listing 2. Zahlen raten

```
10 PRINT"**** KODIEREN/DEKODIEREN ****            <206>
100 GET W$:IF W$="" THEN 100                      <160>
110 IF W$=CHR$(133) THEN 270                      <068>
120 PRINT W$;                                     <065>
130 A$=A$+W$                                      <248>
140 IF W$=CHR$(13) THEN 170                       <142>
150 GOTO 100                                      <078>
160 REM------------------------------             <253>
170 X=X+1                                         <198>
180 A=ASC(MID$(A$,X,1))                           <070>
190 B=A+X                                         <219>
200 B$=CHR$(B)                                    <105>
210 C$=C$+B$                                      <123>
220 IF X<>LEN(A$)-1 THEN 170                      <233>
230 C$=C$+CHR$(32)                                <011>
240 PRINT C$                                      <178>
250 A$=""                                         <023>
260 X=0:GOTO 100                                  <208>
265 REM------------------------------             <254>
270 PRINT"DEKODIEREN"                             <014>
280 INPUT E$                                      <156>
285 IF E$="ENDE" THEN END                         <191>
290 X=X+1                                         <064>
300 E=ASC(MID$(E$,X,1))                           <212>
310 D=E-X                                         <189>
320 D$=D$+CHR$(D)                                 <246>
330 IF X<>LEN(E$) THEN 290                        <117>
340 D$=D$+CHR$(32)                                <159>
350 PRINT D$                                      <042>
360 E$=""                                         <151>
370 X=0:GOTO 280                                  <199>
```

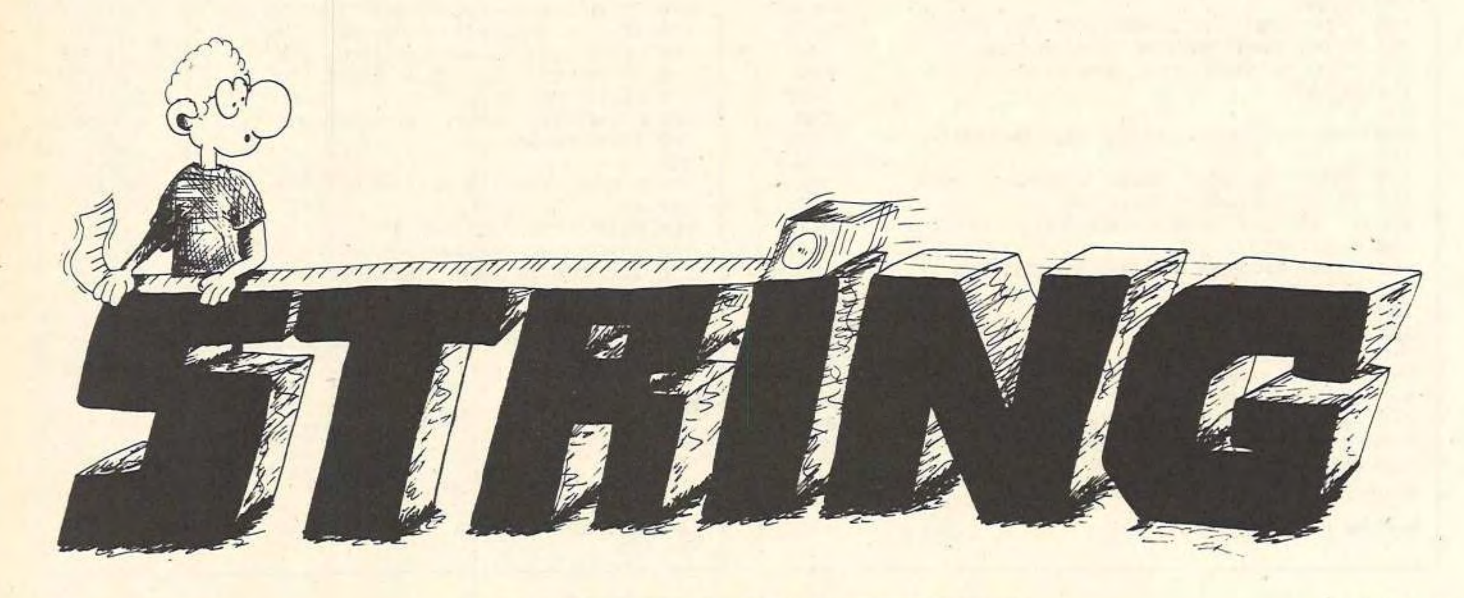
© 64'er

Listing 3. codieren/Decodieren

```
10 PRINT CHR$(147)                                <039>
20 PRINT"***** BEGRIFFE RATEN ******              <211>
30 PRINT                                          <132>
40 :                                              <016>
50 REM-------- SPIELANLEITUNG----------           <206>
60 :                                              <036>
70 :                                              <046>
80 PRINT"DER COMPUTER DENKT SICH EINE             <155>
90 PRINT"SPORTART.                                <175>
100 PRINT"SIE SOLLEN SIE ERRATEN.                 <138>
110 PRINT                                         <212>
120 PRINT"SIE KOENNEN EINEN BUCHSTABEN            <097>
130 PRINT"EINGEBEN, DANN ERSCHEINEN               <187>
140 PRINT"DIE GERATENEN BUCHSTABEN AN             <076>
150 PRINT"DER RICHTIGEN STELLE.                   <165>
160 PRINT                                         <006>
170 PRINT"HABEN SIE DAS WORT ERRATEN,             <021>
180 PRINT"GEBEN SIE ES KOMPLETT EIN.              <169>
190 :                                             <166>
200 :                                             <176>
210 REM------- WORT AUSWAEHLEN -------            <243>
220 :                                             <196>
230 :                                             <206>
240 FOR X=1 TO INT(RND(0)*14)+1                   <155>
250 READ U$                                       <004>
260 NEXT                                          <016>
270 :                                             <248>
280 H$=""                                         <083>
290 FOR X=1 TO LEN(U$)                            <242>
300 H$=H$+"."                                     <008>
310 NEXT                                          <066>
320 :                                             <042>
330 :                                             <052>
340 REM-------- RATEVERSUCH ---------             <160>
350 :                                             <072>
360 PRINT:PRINT                                   <058>
370 PRINT"GESUCHT WIRD: ";H$                      <073>
380 PRINT                                         <228>
390 INPUT"WAS RATEN SIE";L$                       <220>
400 IF L$=U$ THEN 600                             <117>
410 IF LEN(L$)>1 THEN 360                         <067>
420 :                                             <142>
430 :                                             <152>
440 REM--------- HILFSWORT ----------             <065>
450 :                                             <172>
460 :                                             <182>
470 N$=""                                         <041>
480 FOR X=1 TO LEN(U$)                            <176>
490 B$=MID$(U$,X,1)                               <196>
500 C$=MID$(H$,X,1)                               <076>
510 IF L$=B$ THEN N$=N$+L$:GOTO 530               <010>
520 N$=N$+C$                                      <067>
530 NEXT                                          <032>
540 H$=N$                                         <095>
550 GOTO 360                                      <082>
560 :                                             <028>
570 :                                             <038>
580 REM---- NOCH EINMAL ? ----------              <043>
590 :                                             <058>
600 PRINT                                         <194>
610 PRINT"SIE HABEN RICHTIG GERATEN               <116>
620 PRINT                                         <214>
630 PRINT"NOCH EINMAL (J/N) ?                     <096>
640 GET A$: IF A$="" THEN 640                     <166>
650 IF A$="J" THEN RUN                            <166>
660 PRINT                                         <254>
670 PRINT"AUF WIEDERSEHEN                         <006>
680 :                                             <148>
690 :                                             <158>
700 REM------- WOERTERSPEICHER--------            <211>
710 :                                             <178>
720 :                                             <188>
730 DATA FUSSBALL,HOCKEY,GOLF,BOXEN               <115>
740 DATA TURNEN,SCHWIMMEN,HOCHSPRUNG              <099>
750 DATA SEGELN,FECHTEN,JUDO,BASKETBALL           <137>
760 DATA KEGELN,SCHIFAHREN,TENNIS                 <102>
```

© 64'er

Listing 4. Begriffe raten



Sphärenklänge

Wissen Sie eigentlich, welche Klangvielfalt Sie Ihrem C 64 entlocken können; wie man Musik oder Geräusche programmiert? Wir bringen Ihnen eine ausführliche und leicht verständliche Einführung zu diesem interessanten Thema.

Zur Erzeugung von Geräuschen und von Musik ist der C 64 mit einem leistungsfähigen Baustein ausgestattet. Er trägt die Bezeichnung 6581 und soll hier im folgenden SID genannt werden. SID steht für »Sound Interface Device«, was man mit »Klang-Schnittstellen-Baustein« übersetzen könnte. Der SID ist eigentlich ein kleiner Synthesizer, der dreistimmige Melodien spielen oder drei unabhängige Geräusche gleichzeitig erzeugen kann oder auch eine Kombination von beiden, zum Beispiel eine zweistimmige Melodie oder ein Geräusch. Wie man ihn dafür programmiert, soll hier gezeigt werden. Da das Standard-Basic des C 64 keine speziellen Befehle zu diesem Zweck vorsieht, muß man sich näher mit dem inneren Aufbau des SID befassen, um ihn dann mit PEEK- und POKE-Befehlen zu steuern. Dieser gezwungenermaßen etwas unelegante Programmierstil hat aber wenigstens einen Vorteil für denjenigen, der in Maschinensprache programmieren kann oder es lernen will. Er kann nämlich die PEEK- und POKE-Befehle direkt in die Assemblersprache übernehmen. Stürzen wir uns also gleich mittenhinein in die SID-Programmierung (in Basic).

Beim SID wird ein Klang durch folgende Parameter (= Steuergrößen) beeinflußt:

1) Lautstärke
2) Hüllkurve — Sie steuert den zeitlichen Lautstärkenverlauf zum Beispiel eines ausklingenden Tones.
3) Kurvenform — Sie ist für den Klangcharakter des Tones verantwortlich.
4) Frequenz — Sie entspricht der Tonhöhe.

Mit Einzelheiten und mit weiteren Parametern zur Klangsteuerung werden wir uns gleich befassen. Zunächst wollen wir aber einmal einen Ton erzeugen, zum Beispiel um zu hören, ob unser Monitor oder Fernseher, der die Töne wiedergeben muß, richtig eingestellt ist (Perfektionisten schließen den C 64 über die Audio/Video-Buchse und ein normales DIN-Überspielkabel an die HiFi-Anlage an).

POKE 54296,15 — stellt den SID auf maximale Lautstärke
POKE 54278,240 — wählt eine einfache Hüllkurve.
POKE 54273,67 — stellt eine Frequenz ein (zirka 1000 Hz).
POKE 54276,17 — wählt eine sogenannte Dreieckskurve und schaltet zugleich den Ton ein (muß immer als Letztes geschehen!).

Jetzt müßte ein Ton hörbar sein, der ähnlich wie bei einem Fernseh-Testbild klingt.

POKE 54276,16 — schaltet den Ton wieder ab.

Der gleiche Ton mit schärferem Klang gefällig?

POKE 54276,33 — wählt eine »Sägezahnkurve«. Diese klingt heller und schärfer als das Dreieck.

Doch anstatt mit geheimnisvollen POKEs zu arbeiten, sollten wir uns doch besser systematisch mit dem SID befassen. Wer aber nur schnell einen Klangeffekt für ein eigenes Programm benötigt und wen die Einzelheiten des SID nicht so sehr interessieren, der kann den systematischen Teil überspringen und gleich bei »Klangeffekte zum Abtippen« weiterlesen.

Unter einem Register versteht man in der Computertechnik einen Speicherplatz, der mit einer besonderen Funktion gekoppelt ist. Diese Speicherplätze sind also nicht dazu da, um Daten darin abzulegen, sondern um eine Funktion auszulösen oder um Informationen über den Zustand eines Bausteins zu bekommen. Man unterscheidet demnach Schreibregister und Leseregister.

Der SID verfügt insgesamt über 25 Schreib- und Leseregister. Auf Bild 1 sind diese in grafischer Form dargestellt. Der SID hat die Basisadresse:
S = 54272 (dezimal) oder $D400 (hexadezimal)

Unter dieser und den 28 folgenden Adressen können die Register des SID angesprochen werden. Wir werden in Zukunft Registeradressen wie in Bild 1 immer in der Form S+n (n = 0 bis 28) angeben, weil diese Schreibweise prägnanter als eine fünfstellige Zahl ist. Es ist empfehlenswert, sich auch in Programmen an diese Vereinbarung zu halten.

Das Registerschema gliedert sich in drei Blöcke:
Der erste Block ist in Wirklichkeit dreimal vorhanden, für jede Stimme einmal. Die sieben Register dieser Blöcke haben also für die drei Stimmen unterschiedliche Adressen, wie links im Schema auch angegeben ist.

Ein Instrument mit 29 Registern

Der zweite Block (S+21 bis S+24) dient hauptsächlich zur zusätzlichen Klangbeeinflussung durch einen Filter. Den Filter werden wir aber erst später behandeln. Aus diesem Block interessiert zunächst nur die rechte Hälfte des Registers S+24, das für die Lautstärke zuständig ist.

Der dritte Block (S+25 bis S+28) besteht aus vier sogenannten »Nur-Lese-Registern«. Aus diesen Registern kann nur gelesen werden, Schreibzugriffe bleiben wirkungslos. Auch diese Register, die Spezialeffekten dienen, interessieren uns zunächst noch nicht.

Ein Register besteht, wie jeder andere Speicherplatz beim C 64 auch, aus einem Byte, beziehungsweise 8 Bit. Man sieht, daß einige Register noch in Felder unterteilt sind. Bei diesen Registern haben einzelne Bits oder Bitgruppen unterschiedliche Bedeutung. Die schraffierten Bereiche kennzeichnen Bits, die keine Funktion im SID haben. Wir werden bald sehen, wie man einzelne Bits innerhalb eines Byte gezielt ansprechen kann. Nun zu den Registern im einzelnen: Es werden beim ersten Block stellvertretend die Register der Stimme 1 (S+0 bis S+6) beschrieben. Die Register für Stimme 2 (S+7 bis S+13) und Stimme 3 (S+14 bis S+20) sind in ihrer Funktion identisch.

Ab hier ist es praktisch, wenn man bei der Lektüre das kleine Programm aus Listing 1 im Computer hat, denn dann kann man die Wirkung der Parameter in den SID-Registern gleich ausprobieren. Die Parameter stehen gut les- und editierbar in den DATA-Zeilen. Das Programm erzeugt nach dem Starten einen Ton bei einem beliebigen Tastendruck. Tasten mit Auto-Repeat-Funktion, wie zum Beispiel die Space-Taste, erzeugen einen Dauerton. Abgebrochen wird das Programm mit der RUN/STOP-Taste. Der letzte Parameter steuert übrigens die Tonlänge durch eine einfache Verzögerungsschleife.

Frequenz S+0 und S+1

Die Frequenz kann beim SID auf 16 Bit genau angegeben werden. Eine 16-Bit-Zahl kann Werte zwischen 0 und 65535 annehmen. Dieser Wert entspricht allerdings nicht der Frequenz in Hz (Hertz = Schwingungen pro Sekunde). Der SID-Wert F zu einer gegebenen Frequenz in Hz errechnet sich nach:

```
F = 17.0284 * Frequenz
```

Der SID-Wert F zum sogenannten Kammerton a mit 440 Hz beträgt also (ganzzahlig gerundet):

```
F = 17.0284 * 440 ≈ 7492
```

Die höchste vom SID erzeugbare Frequenz beträgt dann (gerundet):

```
65535 / 17.0284 ≈ 3849 (Hz)
```

Zum Experimentieren mit Klangeffekten interessiert uns die genaue Frequenz eigentlich gar nicht, für korrekt gestimmte Tonleitern müssen wir sie dagegen kennen. Zunächst wollen wir aber erfahren, wie man den SID mit dem Wert F (Frequenz) programmiert. Diese im Dezimalsystem maximal fünfstellige Zahl wird im Binärsystem durch 16 Bit dargestellt. Da es sich beim C64 um einen 8-Bit-Mikrocomputer handelt, müssen wir diesen Wert in zwei 8-Bit-Hälften, das sogenannte niederwertige und höherwertige Byte, kurz Low-Byte und High-Byte zerlegen. Hier zwei »Rezepte«:

1. Methode (Standard):

```
HI = INT(F/256)
LO = F-256*HI
```

Das ist nichts anderes als eine Division durch 256 mit Rest. HI ist dabei der Quotient und LO der Divisionsrest. Die Werte LO und HI sind beide Byte-Werte und liegen damit im Bereich 0 bis 255. Im Fall F=7492 (entsprechend 440 Hz) ergibt sich zum Beispiel:

```
HI = 29 und LO = 68
```

2. Methode (mit Einschränkungen, aber schneller):

```
HI = F/256
LO = F AND 255
```

Die INT-Funktion zur Berechnung von HI wurde hier gespart. HI kann hier noch Nachkommastellen haben; diese werden aber später von dem noch folgenden POKE-Befehl abgeschnitten. Die Berechnung von LO funktioniert hier nur bei F-Werten im Bereich 0 bis 32767. Die zweite Methode ist nur dann zu empfehlen, wenn es auf Geschwindigkeit ankommt.

Mit den Werten LO und HI müssen wir dann die beiden Register S+0 und S+1 besetzen:

```
POKE S+0,LO
POKE S+1,HI
```

Man kann die Wirkungsweise des High- und Low-Bytes auch als Grob- und Feineinstellung auffassen. Oft genügt für einen Klangeffekt eine grobe Frequenzsteuerung. Man braucht dann nur das High-Byte zu berücksichtigen und kann das Low-Byte ein für allemal zum Beispiel auf 0 setzen.

Pulsweite S+2 und S+3

Der Parameter »Pulsweite« ist nur wirksam, wenn als Kurvenform das Rechteck gewählt wurde. Die Kurvenformen sind in Bild 1 bei Register S+4 grafisch dargestellt und werden im nächsten Abschnitt besprochen. Das Rechteck ist eine Kurvenform, die nur zwischen zwei Werten hin- und herspringt. Ist der obere Wert genauso lang wie der untere, so spricht man von einer symmetrischen Rechteckkurve. Das Verhältnis zwischen der Länge des oberen und des unteren Wertes kann mit dem Parameter »Pulsweite«, im folgenden P genannt, gesteuert werden. P kann Werte von 0 bis 4095 annehmen und wirkt sich auf die Klangfarbe des Tones aus. Das symmetrische Rechteck, das man mit P = 2048 erhält, klingt verhältnismäßig hohl und wird als typischer Rechteckklang bezeichnet. Entfernt man sich mit P von 2048 in Richtung 0 oder 4095, so wird der Klang zunehmend heller und später schnarrend oder zirpend. Maßgeblich ist hierbei nur der Abstand von P zum Mittelwert 2048. So klingt zum Beispiel P = 2048+500 genauso wie P = 2048-500. Bei P=0 und P=4095 wird kein Ton mehr erzeugt.

P ist eine 12-Bit-Größe und muß wie F in ein Low- und ein High-Byte zerlegt werden. Beim High-Byte können dabei nur die unteren vier Bit gesetzt sein. Zu diesem Zweck kann man ohne Einschränkungen die schon beschriebene Methode 2 anwenden:

```
HI = P/256
LO = P AND 255
POKE S+2,LO
POKE S+3,HI
```

Steuerregister S+4

Dieses Register ist für mehrere Funktionen gleichzeitig zuständig:

- Die Wahl der Kurvenform
- Ein- und Ausschalten des Tones
- Spezialeffekte Ringmodulation und Synchronisation
- Reset der Stimme

Zunächst einmal eine Tabelle mit den Funktionen im einzelnen:

Bit	Dezimalwert (POKE...)	Funktion
0	1	GATE schaltet Ton ein und aus
1	2	SYNC Synchronisation (Spezialeffekt)
2	4	RING Ringmodulation (Spezialeffekt)
3	8	TEST Reset
4	16	wählt Dreieckskurve
5	32	wählt Sägezahnkurve
6	64	wählt Rechteckkurve
7	128	wählt Rauschen

Mit einem POKE an die Adresse S+4 werden immer alle acht Bit gleichzeitig beeinflußt. Einen Befehl zum Setzen oder Löschen einzelner Bits gibt es nicht. Man muß sich daher über die gewünschten Werte aller acht Bits im klaren sein, auch wenn man nur ein Bit verändern will. Um den richtigen POKE-Wert zu erhalten, müssen die Wertigkeiten der Bits, die man setzen will, addiert werden. Dazu drei Beispiele:

1) Rechteck wählen und Ton einschalten
   Bits: 6 und 0
   = Byte-Wert: 2↑6+1 = 65
   POKE S+4,65
2) Ton abschalten, Rechteck gewählt lassen
   Bits: 6
   Byte-Wert: 2↑6 = 64
   POKE S+4,64

Anmerkung: Beim Abschalten eines Tons sollte man immer die zuletzt gewählte Kurvenform gewählt lassen, damit der Ton ausklingen kann. Mit POKE S+4,0 (alle Bits rücksetzen) wird der Ton abrupt abgebrochen.

3) Dreieck mit Ringmodulation wählen, Ton einschalten
   Bits: 4, 2 und 0
   Byte-Wert: 2↑4+2↑2+2↑0 = 16+4+1 = 21
   POKE S+4,21

Die Kurvenform

Sie bestimmt die Klangfarbe des Tones. Am vielseitigsten ist das schon besprochene Rechteck, weil man es durch die Pulsweite reichhaltig gestalten kann. Der Sägezahn klingt noch etwas heller und strahlender als das Rechteck. Er eignet sich besonders gut zur Imitation mancher Instrumentenklänge wie Streicher und Blechbläser. Das Dreieck klingt dagegen weich und dumpf und ist bei tiefen Tönen leider leise. Der Klang ist aber bei hohen Tönen sehr angenehm. Rauschen eignet sich für Effekte wie Wind, Düsenlärm, Schüsse, Explosionen und Schlagzeugklänge. Das Rauschen hat zwar keine feste Tonhöhe, doch sein Klangcharakter wird durch den Frequenzparameter entscheidend beeinflußt.

Bild 1. Alle Register des Sound-Chip auf einen Blick

Die Register des SID — Basisadresse des SID: S = 54272

Adressen: Stimme 1	Stimme 2	Stimme 3	Bitnummern 7	6	5	4	3	2	1	0
S+0	S+7	S+14	Frequenz-low							
S+1	S+8	S+15	Frequenz-high							
S+2	S+9	S+16	Pulsweite-low							
S+3	S+10	S+17					Pulsweite-high (4 Bit)			
S+4	S+11	S+18					Test	Ringmod	Sync	Gate
S+5	S+12	S+19	Attack				Decay			
S+6	S+13	S+20	Sustain				Release			
		S+21						Filterfrequenz-low		
		S+22	Filterfrequenz-high							
		S+23	Resonanz				Filter Ex	Filter 3	Filter 2	Filter 1
		S+24	S3 Aus	Hoch	Band	Tief	Lautstaerke			
		S+25	Potentiometer X							
		S+26	Potentiometer Y							
		S+27	Oszillator 3							
		S+28	Huellkurve 3							

Indem man zwei Kurvenform-Bits gleichzeitig setzt, kann man durch Kombination mehrerer Kurvenformen weitere Klänge erzeugen. Dabei werden die Einzelklänge nicht etwa einfach gemischt, sondern es werden andere Kurvenformen erzeugt. Rauschen läßt sich allerdings nicht mit einer anderen Kurvenform kombinieren. Auch die Kombination von drei Kurvenformen ist unbrauchbar. Sie liefert einen leisen, fast im Rauschen untergehenden Klang. Es bleiben also drei Kombinationen, die sehr interessant klingen:

	POKE-Wert (An/Aus)
Rechteck - Sägezahn	97/96
Rechteck - Dreieck	81/80
Sägezahn - Dreieck	49/48

Der Klangcharakter variiert stark von tiefen zu hohen Tönen. Die letzte Kombination liefert zum Beispiel nur bei sehr tiefen Tönen gute Resultate. Der Klang der ersten beiden Kombinationen hängt natürlich auch von der Pulsweite P ab.

Die Spezialeffekte

Sie sollen hier nur am Rande erwähnt werden. Wird das SYNC-Bit für Stimme 1 gesetzt (Bit 1 in Register S+4), so kann Stimme 1 nicht mehr frei schwingen, sondern wird von Stimme 3 mit beeinflußt, man sagt hier »synchronisiert«. Auch das Ring-Bit bewirkt, daß Stimme 3 die Stimme 1 beeinflußt. Diese sogenannte Ringmodulation wirkt allerdings nur auf die Dreieckskurve. Der Effekt ist daher nur hörbar, wenn das Ring-Bit (Bit 2) zusammen mit Bit 4 für Dreieck gesetzt wird. Beide Effekte liefern ähnliche Resultate. Es lassen sich unter anderem metallische und glockenähnliche Klänge erzeugen. Die Stimme 3 braucht dabei nicht über ihr GATE-Bit eingeschaltet werden. Maßgeblich ist nur die Frequenz von Stimme 3 (Register S+14 und S+15).

Nun besitzen natürlich auch Stimme 2 und 3 je ein SYNC- und ein RING-Bit. Die drei Stimmen steuern sich dabei nach dem Schema:

Stimme 1 ——> Stimme 2
Stimme 2 ——> Stimme 3
Stimme 3 ——> Stimme 1

Das TEST-Bit wird man wahrscheinlich nie benötigen. Es übt eine lokale Reset-Funktion auf die jeweilige Stimme aus. Solange es gesetzt ist, ist nichts hörbar, unabhängig von den anderen Bits. Wenn man allerdings versucht, Rauschen mit einer anderen Kurvenform zu kombinieren, kann es passieren, daß die betroffene Stimme gewissermaßen »abstürzt« und nichts mehr von sich gibt. Man kann sie dann mit einem gezielten Reset über das TEST-Bit wieder zum Leben erwecken.

Die Hüllkurven S+5 und S+6

Wenn eine Stimme über das GATE-Bit eingeschaltet wird, dann folgt ihr zeitlicher Lautstärkenverlauf einer programmierbaren Hüllkurve. Die Hüllkurve bestimmt unter anderem, ob der Ton hart oder weich einsetzt und ob er schnell oder langsam ausklingt. Der Name kommt von den vier Phasen, die die Hüllkurve durchläuft. Jeder Phase ist dabei ein Parameter zugeordnet.

Attack Die Attack-Phase wird durch das Setzen des GATE-Bits eingeleitet. Der Pegel steigt dabei von 0 bis Maximum (Lautstärkeregister) an. Die Zeit für diesen Anstieg ist über den Parameter A in 16 nicht-linearen Stufen von 2 ms bis 8 s einstellbar. Eine kurze Attack-Phase bewirkt einen unmittelbaren und harten Toneinsatz wie bei Schlag- oder Zupfinstrumenten. Eine mittlere Attack-Zeit ist typisch für Bläser- und Streicherklänge, und mit einer langen Attack-Zeit kann man einen Ton wie am Mischpult langsam einblenden.

Decay Nachdem der Maximalwert erreicht ist, fällt der Pegel in der Decay-Phase bis auf den Sustain-Pegel ab. Die Zeit dazu ist mit dem Parameter D in 16 nicht-linearen Stufen von 6 ms bis 24 s einstellbar.

Sustain nennt man die Phase nach dem Pegelabfall in der Decay-Phase. Der Ton klingt dann solange auf dem Sustain-Pegel weiter, bis das GATE-Bit zurückgesetzt wird. Der Parameter SU bestimmt hier also keine Zeit, sondern einen Pegel und zwar in 16 linearen Stufen von Null bis Maximum.

Release Beim Rücksetzen des GATE-Bits wird der Ton nicht einfach abgeschaltet, sondern nimmt in der Release-Phase gleichmäßig vom Sustain-Pegel bis nach Null ab. Die Zeit dazu ist in der gleichen Abstufung wie die Release-Zeit über den Parameter R einstellbar.

Sonderfälle

Wenn das GATE-Bit bereits vor Erreichen der Sustain-Phase rückgesetzt wird, dann startet die Release-Phase mit dem aktuellen Pegel der Hüllkurve. Auf diese Weise ergeben sich:

a) Attack-Decay-Release-Zyklen ADR oder gar nur
b) Attack-Release-Zyklen AR

Bei einem Sustainpegel von Null sind Decay und Release funktionell gleichwertig (Beispiel d in Bild 2). Bei einem maxi-

malen Sustain-Pegel entfällt die Decay-Phase (Attack-Sustain-Release-Zyklus ASR, Beispiel c).

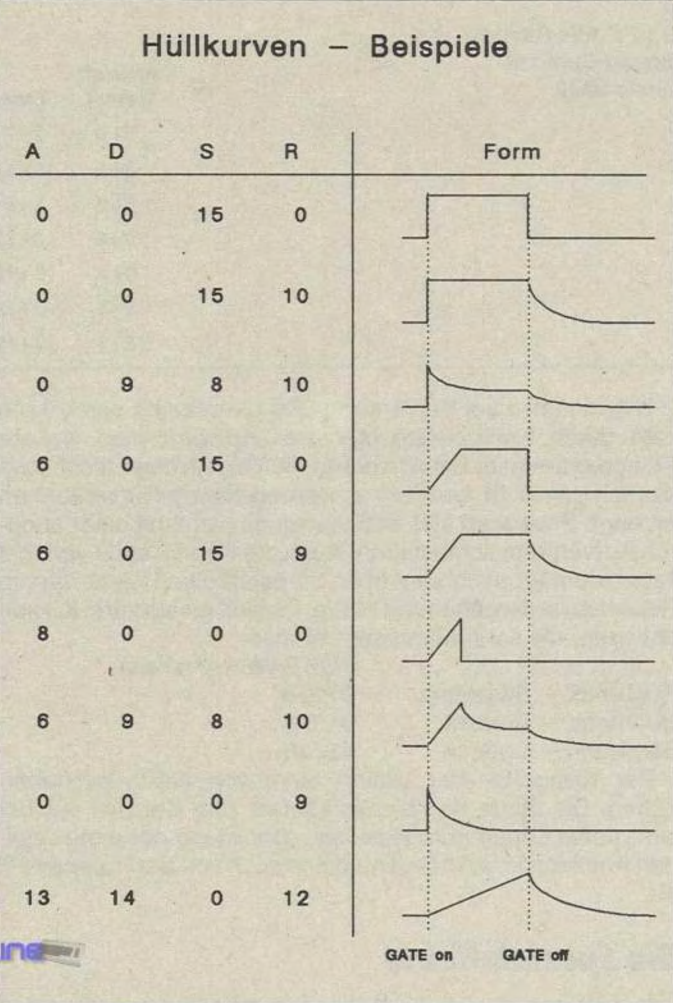
Die Parameter A, D, SU und R sind 4-Bit-Werte. Jeweils zwei von ihnen werden wie folgt in ein Register gepackt:

```
POKE S+5,16*A+D
POKE S+6,16*SU+R
```

Bild 3 zeigt einige Hüllkurvenbeispiele.

Filter, Register S+21 bis S+24

Der Filter bietet neben der Wahl der Kurvenform eine weitere Möglichkeit zur Klangbeeinflussung. Im Gegensatz zu den vorher beschriebenen Parametern muß man sich aber nicht um den Filter kümmern, da er abschaltbar ist und weil der SID auch ohne Filter reichhaltige Klänge erzeugen kann. Um die Wirkungsweise eines Filters zu verstehen, muß man sich einen Klang aus mehreren sogenannten Sinustönen zusammengesetzt denken, dem Grundton und den Obertönen. Sinustöne sind gewissermaßen die nicht mehr weiter zerlegbaren Atome in der Akustik. Ein reiner Sinuston klingt dumpf und ohne charakteristische Färbung. Die vom SID erzeugte Dreiecksschwingung kommt vom Klang her einem Sinuston recht nahe. Die anderen Kurvenformen verdanken ihren helleren Klang einem reichhaltigeren Obertonspektrum (= Folge von Obertönen). Und dieses Obertonspektrum kann man mit dem Filter verändern. Der Filter im SID kennt dazu drei Betriebsarten, die über die Bits 4, 5 und 6 in Register S+24 gewählt werden:



Hüllkurven – Beispiele

A	D	S	R	Form
0	0	15	0	
0	0	15	10	
0	9	8	10	
6	0	15	0	
6	0	15	9	
8	0	0	0	
6	9	8	10	
0	0	0	9	
13	14	0	12	

Bild 2. Die Bedeutung der ADSR-Hüllkurve

Bild 3. Einige ADSR-Werte und die dazugehörigen Hüllkurven

Tiefpaß — Frequenzen beziehungsweise Obertöne oberhalb der in den Registern S+21 und S+22 einstellbaren Filterfrequenz werden abgeschwächt, und zwar umso mehr, je höher diese Frequenzen sind. Der Gesamtklang wird dadurch dunkler und weicher.

Hochpaß — Es werden die Frequenzen abgeschwächt, die unterhalb der Filterfrequenz liegen. Höhere Frequenzen werden ungehindert durchgelassen. Mit einem Hochpaß kann man den Grundton eines Klanges abschwächen. Seine Gesamtzusammensetzung verschiebt sich dann zugunsten der Obertöne. Der Klang wird dabei dünner und heller.

Bandpaß — Dieser Filtermodus schwächt Frequenzen auf beiden Seiten der Filterfrequenz ab. Der Klang wird dabei, wie man fast erwarten kann, etwas dürftig, sofern man nicht maximale Resonanz (siehe weiter unten) einstellt.

Filterfrequenz (Register S+22 und S+23)

Die Filterfrequenz kann auf elf Bit genau eingestellt werden. Dabei kann man aber die drei niederwertigen Bits in Register S+21 unberücksichtigt lassen, da ihr Einfluß praktisch unhörbar ist.

Resonanz und Stimmen-Wahlschalter (Register S+23)

Über den 4-Bit-Parameter Resonanz kann ein gefilterter Klang effektvoller gestaltet werden. Bei großer Resonanz (der Maximalwert ist 15) werden Frequenzanteile in der Gegend der Filterfrequenz verstärkt. Die sonstigen abschwächenden Eigenschaften von Tief- und Hoch- und Bandpaß bleiben dabei erhalten.

Über die Bits 0, 1 und 2 desselben Registers kann man für jede der drei SID-Stimmen unabhängig wählen, ob sie gefiltert oder ungefiltert erklingen soll. Ist zum Beispiel Bit 0 gesetzt, so wird Stimme 1 gefiltert. Das Bit 3, Filter Ex, steuert die Verarbeitung einer von außen zuführbaren Signalquelle, zum Beispiel eines zweiten SID, was uns aber hier nicht kümmern soll.

Filtermodus und Lautstärke (Register S+24)

Die Lautstärkeneinstellung haben wir schon kennengelernt. Man wird sie meistens auf ihren Maximalwert 15 stellen, weil dann der Rauschabstand und damit die Klangqualität am besten ist. Durch Setzen der Bits 4, 5 und 6 wird die Betriebsart des Filters, Tief-, Band- oder Hochpaß gewählt. Die Betriebsarten sind uneingeschränkt kombinierbar. Mit Bit 7 (S3 Aus) kann man die Stimme 3 unhörbar machen. Der Sinn dieser Sonderfunktion wird im Abschnitt über die Leseregister klar.

Ein Beispiel zur Filterprogrammierung:
Stimme 1 soll mit maximaler Resonanz durch den Tiefpaßfilter geschickt werden:

```
FF = 50      Filterfrequenz (nur High-Byte)
FR = 241     (=15*16 für Resonanz +1 für Bit 1)
ML = 31      (=16 für Bit 4 + 15 für Lautstärke)
POKE S+22,FF
POKE S+23,FR
POKE S+24,ML
```

Die Leseregister S+25 bis S+28

Die Register S+25 und S+26 haben mit der Klangprogrammierung nichts zu tun. Über sie können die Werte zweier an Joystick-Ports angeschlossener Potentiometer (Paddles) abgefragt werden. Interessant sind die Register S+27 und S+28:

Aus S+27 kann man den Signalverlauf von Stimme 3 in Form von Byte-Werten lesen. Mit folgendem kleinen Programm kann man diesen Signalverlauf sogar sichtbar machen:

```
10 S=54272
20 POKE S+14,10          :REM F LOW
30 POKE S+15,0           :REM F HIGH
40 POKE S+18,16          :REM DREIECK
50 PRINT TAB(PEEK(S+27)/7);"*":GOTO 50
```

Hier sollte man einmal ein wenig mit den Parametern in Zeile 20-40 experimentieren.

Auf die gleiche Weise kann man aus Register S+28 den Hüllkurvenverlauf von Stimme 3 lesen.

```
100 S=54272
110 POKE S+19,16+11+11      :REM A D
120 POKE S+20,16*8 +11      :REM S R
130 POKE S+18,1             :REM GATE ON
140 FOR I=1 TO 50
150 PRINT TAB(PEEK(S+28)/7);"*"
160 NEXT I
170 POKE S+18,0             :REM GATE OFF
180 FOR I=1 TO 50
190 PRINT TAB(PEEK(S+28)/7);"*"
200 NEXT I
```

Diese Werteverläufe sind besonders zum Modulieren anderer Stimmen geeignet. Normalerweise möchte man Stimme 3 dann nicht hören, wenn man ihren Signalverlauf oder ihre Hüllkurve anderweitig verwendet. Man kann sie dann, wie schon erwähnt, über Bit 7 in Register S+24 ausschalten.

Register	A S+5	D S+5	SU S+6	R S+6	C S+4	P S+2 S+3	F S S+1	FF S+22	FR S+23	ML S+24	M
Glöckchen	0	10	0	10	16	x	40000	x	0	15	100
Oboe	8	7	10	8	64	250	7500	x	0	15	500
Fagott	8	7	10	8	64	250	2500	x	0	15	750
Zungenpfeife	8	0	15	10	48	x	400	x	0	15	1000
Banjo	0	8	0	8	32	x	7500	50	241	111	30
Stahl	0	0	15	12	96	2044	30000	x	0	15	100
Feder	0	8	0	9	32	x	750	x	0	15	35
Preßlufthammer	0	0	15	10	80	2100	200	x	0	15	2000
Schuß	0	8	0	10	128	x	10000	x	0	15	50
Starkstrom	0	0	15	0	128	x	100	x	0	15	2000
Düsenflugzeug	0	0	15	13	128	x	3000	50	241	31	3000
Rakete	0	0	15	15	128	x	1000	10	241	31	3000

x = don't care (Parameter muß nicht eingestellt werden)

Tabelle 1. Einige Beispiel-Effekte für Listing 1

Fest eingestellt: Register	FF S+22				FR S+23			ML S+24		
	x				0			15		
Register	A S+5	D S+5	SU S+6	R S+6	C S+4	P S+2 S+3	F S S+1	G	N	M
Telefon	0	10	0	10	16	x	16000	1.33	2	25
Laserkanone	0	0	15	0	64	1000	30000	0.85	10	1
Take-Off	0	0	15	15	128	x	500	1.004	1000	1
Turbine	0	0	15	15	96	2044	20000	1.001	460	1
Trommelwirbel	0	5	2	9	128	x	20000	1	2	30
Maschinengewehr	0	5	2	9	128	x	12000	0.7	3	30

x = don't care (Parameter muß nicht eingestellt werden)

Tabelle 2. Einige Beispiel-Effekte für Listing 2

Klangeffekte zum Abtippen

Nach diesem systematischen Teil folgen noch Einstellungen. Tabelle 1 enthält einige Parametersätze für Klänge, die der SID ohne großen Programmieraufwand erzeugen kann. Die Klangbezeichnungen wollen die Effekte nur subjektiv beschreiben und sind natürlich nicht zu wörtlich zu nehmen. Man muß nun lediglich die Werte einer Zeile in die, in der Kopfzeile angegebenen SID-Register schreiben, das GATE-Bit setzen und nach einiger Zeit zurücksetzen. Der Parameter M ist übrigens kein SID-Parameter, sondern soll eine Verzögerungsschleife steuern, die die Zeit zwischen GATE ON und GATE OFF bestimmt. Parameter M bezieht sich auf das Programm in Listing 1, das beim Experimentieren Hilfestellung leisten soll. Im DATA-Teil ab Zeile 320 sind die Parameter aus Tabelle 1 einzusetzen und zwar genau in der gleichen Reihenfolge. Das Programm belegt nach dem Start den SID mit den Parametern aus den DATA-Zeilen und wartet auf einen beliebigen Tastendruck, der dann den Klangeffekt auslöst. Man versuche es auch einmal mit den Tasten, die eine Auto-Repeat-Funktion haben, wie zum Beispiel die Space-Taste.

Das Programm aus Listing 2 ist ganz ähnlich aufgebaut, kann aber ein viel größeres Spektrum von Effekten dadurch realisieren, daß es die Frequenz von Stimme 1 dynamisch verändert. Dazu dient die innere Schleife, Zeile 260-300. Dort wird bei jedem Durchlauf die Frequenz F1 mit einem Faktor G multipliziert. Für G>1 steigt die Frequenz schneller an, für G<1 nimmt sie ab und zwar um so schneller, je weiter G von 1 entfernt ist. Die Umrechnung von F1 in Low- und High-Byte geschieht hier nach der schnellen Methode 2. Man beachte, daß damit nur F1-Werte bis 32767 verarbeitet werden können, also nur die Hälfte des vollen Frequenzumfangs des SID. In der äußeren Schleife wird der Wert von F1 auf seinen Ausgangswert F zurückgesetzt. Außerdem steuert die äußere Schleife bei jedem Durchlauf einen ADSR-Hüllkurvenzyklus durch GATE-ON-GATE-OFF. Die Zahl N gibt dabei die Anzahl der inneren Schleifendurchläufe an, die Zahl M die der äußeren Schleifendurchläufe. Beispielparameter zu Listing 2 findet man in Tabelle 2.

Das dritte Programm (Listing 3) verwendet schließlich den Signalverlauf oder die Hüllkurve von Stimme 3, um Stimme 1 zu modulieren. Dies geschieht in der inneren Schleife, Zeile 360-380. Q ist dabei die Adresse eines der Leseregister des SID, also entweder S+27 für den Signalverlauf oder S+28 für den Hüllkurvenverlauf. In der DATA-Zeile 650 ist dazu nur 27 oder 28 anzugeben. Über die Variable G kann die Stärke der Modulation, die sogenannte Modulationstiefe, gesteuert werden. Ein kleinerer Wert von G ergibt hier eine stärkere Modulation. Die äußere Schleife steuert hier ADSR-Hüllkurvenzyklen für Stimme 1 und Stimme 3. Dieses kleine Programm sollte Anlaß zum weiteren Experimentieren sein. Man kann statt der Tonfrequenz zum Beispiel auch einmal versuchen, die Pulsweite oder die Filterfrequenz zu modulieren. Tabelle 3 enthält einige Parametersätze für dieses Programm.

Ausblick: Programmierung von Musikstücken

Dieses Thema wollen wir hier nur einmal streifen. Grundlage hierfür ist die genaue Kenntnis der Tonleiterfrequenzen. Diese müssen aber nicht mühsam aus Tabellen abgetippt, sondern können durch ein kleines Programm selbst berechnet werden. Man muß dazu ein klein wenig Mathematik betreiben:

1) Die Frequenzen zweier Töne im Oktavabstand verhalten sich wie 2:1.
2) Eine Oktave ist durch die Halbtöne in zwölf gleiche Intervalle eingeteilt und zwar nicht linear, sondern exponentiell.
3) Das Frequenzverhältnis H zweier aufeinanderfolgender Halbtöne ist die zwölfte Wurzel aus zwei. In Basic läßt sich das leicht ausrechnen:

```
H = 2(1/12)
```

Register	A	D	SU	R	C	P	F	G	N	M	A3 S+19	B3 S+19	F3 S+20	R3 S+20	C3 S+18	P3 S+16 S+17	F3 S+14 S+15	Q
Vogelgezwitscher	0	8	0	8	16	x	40000	500	8	10	0	8	0	0	x	x	x	28
Bongo	0	7	0	7	16	x	4000	1000	4	1	0	8	0	0	x	x	x	28
E-Baß	0	8	0	9	32	x	750	1000	7	1	0	10	0	0	x	x	x	28
Dampfhammer	0	9	0	11	128	x	5000	200	20	15	0	9	0	9	x	x	x	28
Martinshorn	0	0	15	8	64	1000	7000	720	300	1	x	x	x	x	64	2048	15	27
Sirene	10	13	0	0	64	2048	10000	400	300	1	x	x	x	x	16	x	30	27
Geklimper	0	0	15	0	64	2048	10000	200	400	1	x	x	x	x	128	x	20	27
Grollen	9	10	0	0	32	x	50000	2000	40	10	x	x	x	x	128	x	500	27

Tabelle 3. Einige Beispiel-Effekte für Listing 3

4) Man bekommt dann alle Halbtöne, indem man die Frequenzen, ausgehend von einem Grundwert, fortlaufend mit H multipliziert.

Im Programm aus Listing 4 macht das eine Schleife in Zeile 140 bis 190. Die Variable FAUS wird mit 110 vorbesetzt. Das entspricht einem »großen A«, dem Ton, der zwei Oktaven unter dem Kammerton, dem bekannten »eingestrichenen a« mit 440 Hz, liegt. In Zeile 170 wird FAUS in den dazugehörenden SID-Wert F umgerechnet. F wird in High- und Low-Byte zerlegt, welche in den Feldern FH und FL gespeichert werden. Dort stehen die Töne als fertige POKE-Werte zum Spielen einer Melodie zur Verfügung.

Das restliche Programm erzeugt aus diesen Werten eine mehr oder weniger zufällige Tonfolge auf dem SID. Ein wenig wird der Zufall aber durch die Werte in den DATA-Zeilen ab Zeile 500 gesteuert. Diese Werte stellen ein Blues-Schema in codierter Form dar. Das Schema selbst steht in den letzten drei Zeilen. Das Programm holt sich aus diesem Schema die erste Zahl. Diese zeigt auf eine der sieben Auswahlmengen in den vorausgehenden Zeilen. Die Auswahlmengen enthalten Tonnummern. Das Programm spielt nun nacheinander acht zufällig ausgewählte Töne aus dieser Menge. Tonwiederholungen sind dabei möglich. Anschließend geht das Programm über den nächsten Zeiger aus dem Schema zu einer neuen Auswahlmenge über. Die links etwas abgesetzten Zahlen dienen dabei nur zur Längenangabe der Auswahlmengen und des Schemas.

Mit dieser Technik kann man zwar noch keinen brauchbaren Blues »erwürfeln«, aber der Blues-Charakter wird aus den erzeugten Tonfolgen doch deutlich hörbar.

(Thomas Krätzig/tr)

Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel {CLR}, muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksummer 64 auf Seite 135.

```
100 REM--------------------------------     <146>
110 REM EINFACHE KLANGEFFEKTE               <107>
120 REM--------------------------------     <166>
130 S=54272                                 <150>
140 READ A,D,SU,R,C,P,F,M,FF,FR,ML          <187>
150 POKE S+5 ,16*A +D                       <174>
160 POKE S+6 ,16*SU+R                       <037>
170 POKE S+2 ,P AND 255                     <085>
180 POKE S+3 ,P/256                         <133>
190 HI=INT(F/256):LO=F-256*HI               <142>
200 POKE S   ,LO                            <222>
210 POKE S+1 ,HI                            <018>
220 POKE S+22,FF                            <056>
230 POKE S+23,FR                            <154>
240 POKE S+24,ML                            <224>
250 GET A$:IF A$="" THEN 250                <220>
260 : POKE S+4,C OR 1                       <191>
270 : FOR I=1 TO M:NEXT I                   <161>
280 : POKE S+4,C AND 254                    <193>
290 GOTO 250                                <052>
300 REM--------------------------------     <092>
310 REM PARAMETER                           <224>
320 DATA 0 ,10, 0,10:REM A D S R            <165>
330 DATA 16         :REM CONTROL-BYTE       <222>
340 DATA 2048       :REM PULSWEITE          <129>
350 DATA 40000      :REM FREQUENZ           <183>
360 DATA 100        :REM VERZOEGERUNG       <216>
370 DATA 50         :REM FILTERFREQUENZ     <033>
380 DATA 0          :REM FILTERRESONANZ     <148>
390 DATA 15         :REM MODUS/LAUT         <209>
```

Listing 1. Einfache Klangeffekte. Hinweise im Text.

```
100 REM-----------------------------------    <146>
110 REM KLANGEFFEKTE MIT                       <098>
120 REM DYNAMISCHER FREQUENZSTEUERUNG          <218>
130 REM-----------------------------------    <176>
140 S=54272                                    <160>
150 READ A,D,SU,R,C,P,F,G,N,M                  <206>
160 POKE S+5 ,16*A +D                          <184>
170 POKE S+6 ,16*SU+R                          <047>
180 POKE S+2 ,P AND 255                        <095>
190 POKE S+3 ,P/256                            <143>
200 POKE S+23,0        :REM FR                 <203>
210 POKE S+24,15       :REM ML                 <057>
220 GET A$:IF A$="" THEN 220                   <253>
230 FOR I=1 TO M                               <086>
240 : F1=F                                     <004>
250 : POKE S+4,C OR 1                          <179>
260 : FOR J=1 TO N                             <224>
270 :   POKE S,F1 AND 255                      <039>
280 :   POKE S+1,F1/256                        <178>
290 :   F1=F1*G                                <010>
300 : NEXT J                                   <206>
310 : POKE S+4,C                               <204>
320 NEXT I                                     <150>
330 GOTO 220                                   <044>
400 REM-----------------------------------    <192>
410 REM PARAMETER                              <068>
420 DATA 0 ,10, 0,10:REM A D SU R              <088>
430 DATA 16          :REM CONTROL-BYTE C       <092>
440 DATA 2048        :REM PULSWEITE P          <038>
450 DATA 16000       :REM FREQUENZ F           <124>
460 DATA 1.33        :REM FAKTOR G             <117>
470 DATA 2           :REM ANZAHL N             <197>
480 DATA 25          :REM ANZAHL M             <084>
```

Listing 2. Klangeffekte mit Frequenzsteuerung

```
100 REM-----------------------------------    <146>
110 REM KLANGEFFEKTE MIT                       <098>
120 REM DYNAMISCHER STEUERUNG                  <139>
125 REM DURCH STIMME 3                         <066>
130 REM-----------------------------------    <176>
140 S=54272                                    <160>
150 READ A,D,SU,R,C,P,F,G,N,M                  <206>
160 POKE S+5 ,16*A +D                          <184>
170 POKE S+6 ,16*SU+R                          <047>
180 POKE S+2 ,P AND 255                        <095>
190 POKE S+3 ,P/256                            <143>
200 POKE S+23,0        :REM FR                 <203>
210 POKE S+24,128+15:REM ML (S3 AUS)           <086>
220 READ A3,D3,S3,R3,C3,P3,F3,Q                <190>
230 HI=INT(F3/256):LO=F3-256*HI                <201>
240 POKE S+14,LO                               <197>
250 POKE S+15,HI                               <255>
260 POKE S+16,P3 AND 255                       <111>
270 POKE S+17,P3/256                           <209>
280 POKE S+19,16*A3+D3                         <209>
290 POKE S+20,16*S3+R3                         <204>
300 Q=S+Q                                      <230>
310 F=F/256                                    <091>
320 GET A$:IF A$="" THEN 320                   <131>
330 FOR I=1 TO M                               <188>
340 : POKE S+4 ,C OR 1                         <015>
350 : POKE S+18,C3 OR 1                        <012>
360 : FOR J=1 TO N                             <068>
370 :   POKE S+1,F*(1+PEEK(Q)/G)               <106>
380 : NEXT J                                   <030>
390 : POKE S+4 ,C                              <028>
400 : POKE S+18,C3                             <166>
410 NEXT I                                     <240>
420 GOTO 320                                   <142>
500 REM-----------------------------------    <036>
510 REM PARAMETER STIMME 1                     <250>
520 DATA 0 , 8, 0, 8:REM A D SU R              <243>
530 DATA 32          :REM CONTROL-BYTE C       <146>
540 DATA 2048        :REM PULSWEITE P          <140>
550 DATA 40000       :REM FREQUENZ F           <154>
560 DATA 500         :REM FAKTOR G             <239>
570 DATA 8           :REM ANZAHL N             <091>
580 DATA 10          :REM ANZAHL M             <098>
600 REM PARAMETER STIMME 3                     <094>
610 DATA  0, 8, 0, 0:REM A3 D3 S3 R3           <144>
620 DATA 16          :REM CONTROL C3           <233>
630 DATA 2048        :REM PULSWEITE P3         <127>
640 DATA 10          :REM FREQUENZ F3          <063>
650 DATA 28          :REM MOD.-QUELLE Q        <201>
```

Listing 3. Klangeffekte mit Steuerung durch Stimme 3

```
10 REM-----------------------------------     <161>
20 REM ZUFALLSTONFOLGE                         <234>
30 REM MIT BLUES-SCHEMA                        <079>
40 REM                                         <102>
50 REM AUSNUETZUNG ALLER DREI STIMMEN          <099>
60 REM ZUR KLANGVERBESSERUNG                   <238>
70 REM                                         <132>
80 REM T. KRAETZIG            MAERZ 86         <245>
90 REM-----------------------------------     <241>
100 DIM FL(25) :REM ARRAY F. FREQUENZEN        <058>
101 DIM FH(25)                                 <137>
102 DIM A(8,20):REM AUSWAHLMENGEN              <164>
104 DIM S(30)  :REM SCHEMA                     <131>
110 S  =54272  :REM BASISADRESSE               <235>
130 :                                          <106>
140 REM TONLEITER-FREQUENZEN BERECHNEN         <166>
150 FAUS=110:H=2↑(1/12)                        <026>
160 FOR I=0 TO 25                              <152>
170 :  F=INT(FAUS*17.0284+0.5)                 <123>
172 :  FH(I)=INT(F/256)                        <213>
174 :  FL(I)=F-256*FH(I)                       <097>
180 :  FAUS=FAUS*H                             <067>
190 NEXT I                                     <018>
200 :                                          <176>
210 REM PARAMETER FESTLEGEN                    <133>
220 PW=2048         :REM PULSWEITE             <137>
230 C =32           :REM KURVENFORM            <135>
240 A=0:D=10:SU=0:R=9                          <192>
250 FOR I=0 TO 14 STEP 7                       <159>
255 : POKE S+I+2,PW AND 255                    <128>
260 : POKE S+I+3,PW/256                        <233>
265 : POKE S+I+5,16*A+D                        <160>
270 : POKE S+I+6,16*SU+R                       <024>
275 NEXT I                                     <105>
280 :                                          <002>
290 REM FILTER AUS UND LAUTSTAERKE MAX.        <155>
300 POKE S+23,0:POKE S+24,15                   <066>
310 :                                          <032>
320 REM AUSW.MENGEN UND SCHEMA EINLESEN        <240>
325 READ I:A(0,0)=I                            <245>
330 FOR K=1 TO I                               <202>
335 :  READ J:A(K,0)=J                         <228>
340 :  FOR L=1 TO J:READ A(K,L):NEXT L         <013>
350 NEXT K                                     <196>
355 READ I:S(0)=I                              <175>
360 FOR K=1 TO I                               <232>
365 :  READ S(K)                               <051>
370 NEXT K                                     <216>
375 :                                          <097>
380 REM ZUFALLSTONFOLGE                        <084>
385 L=0:O=0                                    <239>
390 FOR I=1 TO S(0)                            <039>
395 :  J=S(I)                                  <051>
400 :  N=A(J,0)                                <158>
405 :  FOR K=1 TO 8                            <107>
410 :     ZZ=A(J,INT(RND(1)*N+1))              <064>
412 :     POKE S+L  ,FL(ZZ+O)                  <166>
414 :     POKE S+L+1,FH(ZZ+O)                  <115>
416 :     POKE S+L+4,C OR 1                    <105>
420 :     FOR P=1 TO 40:NEXT                   <035>
425 :     POKE S+L+4,C                         <218>
430 :     FOR P=1 TO 40:NEXT                   <045>
435 :     L=L+7:IF L=21 THEN L=0               <121>
440 :  NEXT K                                  <106>
445 NEXT I                                     <019>
450 O=O+1:IF O=4 THEN O=0                      <003>
455 FOR P=1 TO 1150:NEXT                       <183>
460 GOTO 390                                   <038>
500 REM-----------------------------------    <036>
510 REM AUSWAHLMENGEN UND SCHEMA               <016>
520 REM-----------------------------------    <058>
530 DATA  7                                    <221>
540 DATA  8,  0,4,7,10,12,16,19,22             <243>
550 DATA  8,  0,3,5,9, 12,15,17,21             <238>
560 DATA  7,  2,5,7,11,14,17,19                <126>
570 DATA  6,  0,0,4, 7, 7,10                   <096>
580 DATA  4,  0,3,5,9                          <114>
590 DATA  5,  2,5,7,7,11                       <237>
610 DATA  2,  0,7                              <115>
620 :                                          <088>
630 DATA 24,  7,1,2,1,3,2,1,3                  <210>
640 DATA      4,4,5,4,6,5,4,6                  <058>
650 DATA      7,1,2,1,3,2,1,3                  <117>
```

Listing 4. Programm zum Errechnen einer Zufalls-Blues-Musik

Rezepte für Grafik-Diners

Die interessantesten Grafik-Befehle verschweigt Ihnen Ihr Handbuch! Wollen Sie Ihren Computer wirklich kennenlernen? Wollen Sie wissen, wie man seine Fähigkeiten vollständig ausreizt, um die berühmte Grafik-Vielfalt des C64 auf den Schirm zu kriegen? Dann lesen Sie!

Es geht wohl fast jedem Einsteiger so: Ein Grund für den Erwerb des C64 waren die Angaben über die guten Grafik-Eigenschaften dieses Computers. Sind dann die ersten Hürden des Handbuches genommen, entsteht der Wunsch, nun auch mal in schönen Bildern oder exakten Graphen zu schwelgen. Und dann folgt die Enttäuschung: Das Handbuch schweigt sich weitgehend aus, das Basic 2.0 enthält keinerlei Grafik-Befehle! Sie haben nun drei Möglichkeiten, mit dieser Erkenntnis umzugehen: Lassen Sie die Grafik Grafik sein (würde ich nicht empfehlen), kaufen Sie sich eine der vielen auf dem Markt angebotenen Basic-Erweiterungen zur Grafik-Unterstützung (dann haben Sie eine Reihe von neuen Basic-Befehlen zur Verfügung, mit denen Sie beispielsweise zeichnen oder malen können, sind aber andererseits oft eingeschränkt: Ihr Computer kann meistens noch viel mehr!) oder Weg drei: Fassen Sie sich ein Herz und steigen Sie unerschrocken in die Labyrinthe der Ebenen unter der Hochsprache Basic. Das kann ein richtiges Abenteuer werden, die Register des Videochip und anderer Bausteine kennenzulernen und die weiten Ebenen des RAM zu durchstreifen, ungewöhnlichen Zahlen zu begegnen und anderes mehr. Sie brauchen diese Kreuzzüge nicht alleine durchzuführen. Tatkräftige Helfer stehen Ihnen in Form von Literatur zur Seite. Der vorliegende Artikel soll Ihnen in einem Schnellverfahren einfache Rezepte vorstellen, mit deren Hilfe Sie grafische Aufgabenstellungen lösen können. Falls Ihnen dann aber der Sinn nach mehr steht, bedienen Sie sich einiger der nachfolgend vorgestellten literarischen Mitstreiter:

Register	Adresse	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	53248	X-Position des Sprite Nr. 0. Dazu muß Register 16 beachtet werden							
1	53249	Y-Position des Sprite Nr. 0							
2	53250	X-Position des Sprite Nr. 1. Auch dazu, wie zu allen folgenden Sprites, muß Register 16 beachtet werden.							
3	53251	Y-Position des Sprite Nr. 1							
4	53252	X-Position des Sprite Nr. 2. s. o.							
5	53253	Y-Position des Sprite Nr. 2							
6	53254	X-Position des Sprite Nr. 3. s. o.							
7	53255	Y-Position des Sprite Nr. 3							
8	53256	X-Position des Sprite Nr. 4. s. o.							
9	53257	Y-Position des Sprite Nr. 4							
10	53258	X-Position des Sprite Nr. 5. s. o.							
11	53259	Y-Position des Sprite Nr. 5							
12	53260	X-Position des Sprite Nr. 6. s. o.							
13	53261	Y-Position des Sprite Nr. 6							
14	53262	X-Position des Sprite Nr. 7. s. o.							
15	53263	Y-Position des Sprite Nr. 7							
16	53264	Spr. 7, msb X-Pos.	Spr. 6, msb X-Pos.	Spr. 5, msb X-Pos.	Spr. 4, msb X-Pos.	Spr. 3, msb X-Pos.	Spr. 2, msb X-Pos.	Spr. 1, msb X-Pos.	Spr. 0, msb X-Pos.
17	53265	msb des Rasterregisters (Reg. 18)	Schaltbit für veränderten Hintergrundfarbmodus 1 = eingeschaltet	Schaltbit für Hochauflösungsmodus 1 = eingeschaltet	Schaltbit für Bildschirm »aus« 0 = normaler Bildschirm 1 = Bildschirmfarbe gleich Hintergrundfarbe	Schaltbit für Zeilenzahl 0 = 24 Zeilen 1 = 25 Zeilen	Wert der Zeilenverschiebung in Y-Richtung beim Smooth Scrolling		
18	53266	Rasterregister. Dazu kommt das msb in Bit 7, Register 17							
19	53267	Lichtgriffel X-Position							
20	53268	Lichtgriffel Y-Position							
21	53269	Ein- und Ausschalten von Sprites. 0 = Sprite aus. 1 = Sprite an							
		Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
22	53270			Reset-Bit, muß 0 sein, damit VIC-II-Chip arbeitet	Schaltbit für Mehrfarbmodus 1 = eingeschaltet	Schaltbit für Spaltenzahl 0=38 Spalten 1=40 Spalten	Wert der Spaltenverschiebung in X-Richtung beim Smooth Scrolling		

Tabelle 1. Registerübersicht des VIC-II-Chips

1) H. Ponnath, »C64-Wunderland der Grafik«, München 1985: Markt&Technik Verlag, MT 756 (Umfassende Einführung inklusive Diskette mit allen Beispielen, unter anderem auch mit einer Befehlserweiterung für Grafik-Programmierung).
2) S. Krute, »Grafik & Musik auf dem C64«, München 1984: Markt&Technik Verlag, MT 743 (Legt Schwergewicht auf die Sprites und Musik).

Prinzip der Grafikprogrammierung

Bevor wir ans Kochen der verschiedenen Grafik-Gerichte gehen, sollten Sie noch etwas über das Handwerkszeug und die Küche erfahren. Wir sind gezwungen, unterhalb der Ebene der Hochsprache Basic zu operieren. Dafür wäre die geeignete Computersprache eigentlich Assembler. Viele Grafik-Programme bedienen sich daher auch dieser Sprache, was aber für Sie als Einsteiger den Nachteil mit sich brächte, daß Sie zuerst noch Assembler lernen müßten, um zu verstehen, was hier geschieht. Es gibt aber einige Basic-Befehle, die gewissermaßen zwischen den Ebenen verkehren. Dazu gehören vor allem POKE und PEEK. Mittels POKE kann man einen Wert direkt in eine Speicherstelle eintragen, mittels PEEK kann andererseits festgestellt werden, welcher Wert in einer Speicherstelle enthalten ist. Genau das erfordert die Grafik-Programmierung. Bestimmte Werte müssen in Register oder spezielle Speicherbereiche geschrieben oder aus diesen gelesen werden. Wir werden also Basic-Programme mit allerlei solchen POKE- und PEEK-Kommandos verwenden. Was sind Register? Grafikprogrammierung beim C64 ist eigentlich in der Hauptsache die Programmierung eines besonderen Bausteines unseres Computers, nämlich des VIC-II-Chip. VIC ist die Abkürzung von »Video-Interface-Controller«, was bedeutet, daß es sich hierbei um den Baustein handelt, der die Kooperation des Computers mit dem Bildschirm kontrolliert. Die Programmierung eines solchen Chips geschieht fast ausschließlich durch bestimmte Speicherstellen (für den VIC sind das die zwischen 53248 und 53294), die man eben Register nennt. Je nach Inhalt dieser Register nimmt der VIC unterschiedliche Betriebszustände an, und weil mit diesen Registern die Grafikprogrammierung steht oder fällt, finden Sie sie in der Tabelle 1 alle aufgeführt.

Die Möglichkeiten, die uns der VIC-II-Chip grafisch eröffnet, sind immens. Sechs davon werden Ihnen nachfolgend als Kochrezepte vorgestellt:

Register	Adresse	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
23	53271	Sprite-Vergrößerung in Y-Richtung. 0 = normale Größe. 1 = doppelte Größe.							
		Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
24	53272	Startadresse des Bildschirmspeichers				— Startadresse des Speicherbereichs, in dem die Zeichen als Punktmatrizen abzurufen sind. — Startadresse der Bit-Map			
25	53273	Interrupt-Flaggen-Register Interrupt				Lichtgriffel-Interrupt-Flagge	Sprite/Sprite-Kollision	Sprite/Hintergrund-Kollision	Raster-Interrupt-Flagge
26	53274	Interrupt-Masken-Register Interrupt				Lichtgriffel-Interrupt-Maske	Sprite/Sprite-Koll.-Maske	Sprite/Hintergrund-Kollision Maske	Raster-Interrupt-Maske
27	53275	Sprite/Hintergrund-Prioritätenregister. 0 = Sprite hat Priorität. 1 = Hintergrund hat Priorität							
		Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
28	53276	Sprite-Mehrfarbmodus-Register. 0 = Normaldarstellung. 1 = Mehrfarbmodus-Darstellung							
		Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
29	53277	Sprite-Vergrößerung in X-Richtung. 0 = normale Größe. 1 = doppelte Größe							
		Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
30	53278	Sprite/Sprite-Kollision. 0 = keine Berührung. 1 = Berührung							
		Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
31	53279	Sprite/Hintergrund-Kollision. 0 = keine Berührung. 1 = Berührung							
		Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
32	53280	unbenutzt				Farbe des Bildschirmrahmens			
33	53281	unbenutzt				Hintergrundfarbe Nr. 0 (normale Hintergrundfarbe)			
34	53282	unbenutzt				Hintergrundfarbe Nr. 1			
35	53283	unbenutzt				Hintergrundfarbe Nr. 2			
36	53284	unbenutzt				Hintergrundfarbe Nr. 3			
37	53285	unbenutzt				Sprite-Mehrfarben-Register Nr. 0			
38	53286	unbenutzt				Sprite-Mehrfarben-Register Nr. 1			
39	53287	unbenutzt				Sprite 0, Farbe			
40	53288	unbenutzt				Sprite 1, Farbe			
41	53289	unbenutzt				Sprite 2, Farbe			
42	53290	unbenutzt				Sprite 3, Farbe			
43	53291	unbenutzt				Sprite 4, Farbe			
44	53292	unbenutzt				Sprite 5, Farbe			
45	53293	unbenutzt				Sprite 6, Farbe			
46	53294	unbenutzt				Sprite 7, Farbe			

Blockgrafik
Zeichensatzänderungen
mehrfarbige Zeichen
Sprites
hochauflösende Grafik
Mehrfarbengrafik

So! Nun wollen wir die Töpfe, Pfannen und Kochlöffel schwingen zum ersten grafischen Gericht:

Blockgrafik

Unter Blockgrafik versteht man die Art Grafik, die mit den Zeichen realisiert wird, die unsere Tastatur hergibt. Reine Grafikzeichen findet man hier zwischen CHR$(96) und CHR$(127) sowie von CHR$(161) bis CHR$(254). Abgesehen von den Spielkartenzeichen CHR$(97), CHR$(115), CHR$(120) und CHR$(122) oder Bällen CHR$(113), CHR$(119) oder einfach durch eine Reihe von PRINT-Befehlen aufgebaute Abbildungen können durchaus auch ernsthafte und nützliche Dinge mit Blockgrafik betrieben werden. Beispielsweise gibt es vereinzelt in Fachzeitschriften Programme zu Kurvendiskussionen, die einen Kurvenverlauf mittels der sogenannten Viertelblockgrafik zeichnen (dazu gehören beispielsweise die Zeichen CHR$(187) und andere). Manchmal hört man hier auch den Begriff »Grafik mittlerer Auflösung«, weil man bei geschickter Programmierung aus den 40 horizontalen Positionen 80 und aus den 25 vertikalen 50 hervorzaubern kann. Aber auch mit der niedrigen Auflösung von 40 mal 25 Bildschirmpositionen lassen sich sinnvolle Anwendungen realisieren. Wir wollen hier ein Programm zur Erstellung von Säulendiagrammen entwickeln, das bis zu 12 Werte in horizontalen Säulen übersichtlich abbildet (Listing 1).

Hier ein Kommentar zu den Programminhalten:

20 Rahmenfarbe schwarz, Hintergrund grau.
30 Zeichenfarbe grün, Bildschirm löschen.
40 Liste aller Variablen
50 Erzeugung zweier Strings für einen Vertikaltabulator.
DN$ = 24mal Cursor down
UP$ = 24mal Cursor up.
65 Hier gibt es die Möglichkeit, entweder einen eingebauten Test oder aber eigene Daten abbilden zu lassen. Wenn Sie später den Test nicht mehr benötigen, löschen Sie einfach die Zeilen 65, 67 und 400 bis 460.
70 Abfrage auf die Anzahl der Werte. Zulässig ist hier eins bis zehn.
80 Falls die Anzahl außerhalb des erlaubten Bereiches liegt, wird erneut abgefragt.
90 Zwei Felder werden dimensioniert. W(N) enthält die Werte, W$(N) eine Kenn-Bezeichnung für diese Werte.
100 Eine Kopfzeile für die Eingabe der Werte wird gedruckt.
110 Eingabeschleife.
120 Abfrage der Kenn-Bezeichnung. Nur die beiden ersten Zeichen werden gespeichert.
130 Abfrage des Wertes.
140 Jeder Wert wird verglichen mit einem Maximalwert MA. Ist der aktuelle Wert größer als MA, dann wird er zu MA. Am Ende der Eingabeschleife befindet sich der größte Wert in MA.
150 Ende der Eingabeschleife.
160 Hier muß eine Kopfzeile, ein Titel des Diagrammes, eingegeben werden.
170 Falls diese Kopfzeile kürzer als 40 Zeichen ist, wird ihr noch ein Cursor down angehängt.
190 30 horizontale Positionen werden für den Maximalwert reserviert. Jede Position entspricht nun einem Differenzwert DX. Die 22 freien Vertikalen (25-2 für den Kopf und -1 für die letzte Zeile) werden auf die Anzahl der Werte aufgeteilt.
210 Bildschirm rollt hinauf, Cursor in Home-Position, Hintergrund schwarz.
220-240 Eine vertikale Linie erscheint als Grundlinie auf dem Bildschirm.
250 Cursor in Home-Position, Drucken des Titels.
260 Die Zeichenschleife beginnt hier.
265 Der vertikale Tabulator setzt den Cursor entsprechend der Werteanzahl abwärts.
270 Drucken der Kennbezeichnung, Reversmodus einschalten, Zeichenfarbe rot, Linie überspringen.
280-300 Aus reversen Spacezeichen wird in dieser Schleife ein waagerechter Balken gezeichnet. Seine Länge hängt ab vom Verhältnis des aktuellen Wertes zum Differenzwert DX.
310 Reversmodus ausschalten, Zeichenfarbe grün und Drucken des Wertes ab Position 33. Zu beachten ist, daß der Wert nicht mehr als sechs Zif-

```
10 REM ***** PROGRAMM SAEULENDIAGRAMME ***
   **                                      <202>
20 POKE 53280,0:POKE 53281,11              <033>
30 PRINT CHR$(30)CHR$(147)                 <094>
40 I=0:J=0:N=0:DX=0:DY=0:Z=0:MA=0:DN$="":U
   P$="":K$="":A$=""                       <232>
50 FOR I=1 TO 24:DN$=DN$+CHR$(17):UP$=UP$+
   CHR$(145):NEXT I                        <044>
60 REM --- EINGABETEIL ---                 <216>
65 INPUT"TEST(1) ODER EIGENE WERTE(2)";Z   <019>
67 IF Z<>2 THEN GOTO 400                   <069>
70 PRINT CHR$(17)"WIEVIELE WERTE (MAX.10)"
   TAB(30);:INPUT N                        <247>
80 :IF N<1 OR N>10 THEN PRINT CHR$(147):GO
   TO 70                                   <177>
90 DIM W(N),W$(N)                          <028>
100 PRINT CHR$(17)TAB(2)"BEZEICHNUNG (2 ZE
    ICHEN)"TAB(27)"WERT"CHR$(17)           <099>
110 FOR I=1 TO N                           <095>
120 :PRINT I TAB(10);:INPUT W$(I):W$(I)=LE
    FT$(W$(I),2)                           <128>
130 :PRINT CHR$(145)TAB(25);:INPUT W(I)    <213>
140 ::IF ABS(W(I))>MA THEN MA=W(I)         <094>
150 NEXT I                                 <234>
160 PRINT CHR$(17)"KOPFZEILE:":INPUT K$    <110>
170 :IF LEN(K$)<40 THEN K$=K$+CHR$(17)     <189>
180 REM --- BERECHNUNGSTEIL ---            <151>
190 DX=MA/30:DY=INT(22/N)-1                <165>
200 REM --- ZEICHENTEIL ---                <094>
210 PRINT DN$CHR$(19):POKE 53281,0         <174>
220 FOR I=1 TO 23                          <242>
230 :PRINT TAB(2)CHR$(98)                  <010>
240 NEXT I                                 <068>
250 PRINT CHR$(19)K$                       <187>
260 FOR I=1 TO N                           <247>
265 :PRINT LEFT$(DN$,DY);                  <216>
270 :PRINT W$(I);CHR$(18)CHR$(28)TAB(3);   <237>
280 ::FOR J=1 TO INT(W(I)/DX)              <068>
290 ::PRINT CHR$(32);                      <129>
300 :NEXT J                                <206>
310 :PRINT CHR$(146)CHR$(30)TAB(33)W(I)    <251>
330 NEXT I                                 <160>
340 GET A$:IF A$=""THEN 340                <024>
350 END                                    <098>
400 REM ---- TESTLAUF ----                 <227>
410 N=10:DIM W(N),W$(N)                    <230>
420 FOR I=1 TO N:READ W$(I),W(I):NEXT I    <195>
430 MA=875                                 <136>
440 K$="WAFFENEXPORTE DER BUNDESREPUBLIK D
    EUTSCHLAND 1969 BIS 1978 (MIO DOLLAR)" <214>
450 GOTO 170                               <236>
460 DATA 69,100,70,190,71,130,72,330,73,14
    0,74,210,75,420,76,650,77,850,78,875   <163>
```

Listing 1. »Säulendiagramme«

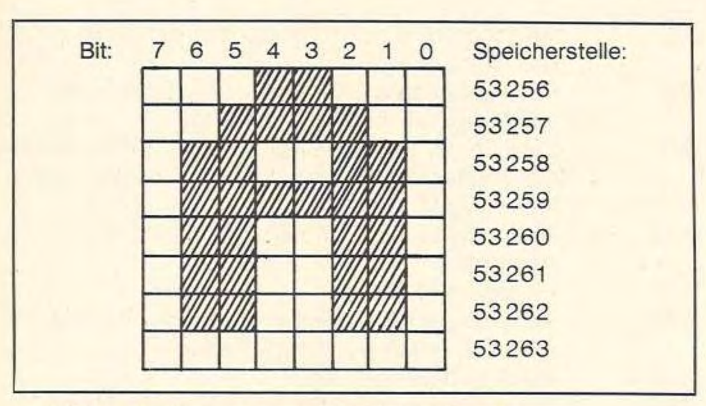


Bild 1. Das Zeichenmuster des Buchstaben A

Bitnummer:	7	6	5	4	3	2	1	0
Wert:	128	64	32	16	8	4	2	1

Bild 2. Die Werte der Bits

fern (inklusive Dezimalpunkt) enthält, weil sonst in der nächsten Zeile weitergeschrieben wird. Hier könnte das Programm noch eine Veränderung ertragen. Probieren Sie es doch mal aus!

330 Ende der Zeichenschleife.

340 Das Bild bleibt bis zu einem Tastendruck eingefroren.

400- Das ist der Eingabeteil für den Fall, daß Sie den Testlauf wählen. Wie Sie – beispielsweise aus dem Kommentar zu Zeile 310 – feststellen können, ist dieses Programm durchaus noch erweiterungsfähig. Zum einen ist es nämlich nicht ganz wasserdicht: Man kann durch bestimmte Eingaben eine Fehlermeldung und den Programmabschluß erzwingen. Das darf bei professioneller Software natürlich nicht passieren. Zum anderen aber könnte man die – jetzt sehr grobe – Darstellung verfeinern, indem man außer den reversen Spaces auch noch andere Zeichen (beispielsweise CHR$(161)) einplant. Daran können Sie sich ja mal erproben. Dieses Rezept verhilft uns zunächst lediglich zu einer leichten Vorspeise. Ein weiteres Gericht hat ebenfalls mit den Zeichen zu tun.

Zeichenmusteränderung

Der Computer hat für jedes Zeichen, das er auf dem Bildschirm zeigen kann, acht Speicherstellen, in denen er sich merkt, wie das Zeichen aussehen soll. All diese Zeichenmuster bewahrt er im sogenannten Zeichen-ROM auf. Beispielsweise ist dort das Zeichen A so gespeichert, wie es Bild 1 zeigt:

Bevor wir weitermachen, müssen noch ein paar Begriffe erklärt werden:

ROM Speicher, aus dem nur gelesen werden kann.
RAM Speicher, der Schreiben und Lesen erlaubt.
Bit Das ist die kleinste Informationseinheit. Ein Bit kann entweder 1 oder 0 enthalten. Man kann sich das vorstellen wie eine Lampe, die entweder angeschaltet (also 1) oder ausgeschaltet (also 0) sein kann.
Byte Acht Bits werden zusammengefaßt und bekommen den Namen Byte. In unserem Computer paßt in jede Speicherstelle genau ein Byte.
Page 256 solcher Bytes nennt man eine Page. Vier Pages nennt man 1 KByte. Das sind dann genau 1024 Bytes. Vorsicht: 1 KByte darf nicht verwechselt werden mit 1 kByte, denn das letztere wird ein Kilobyte genannt und entspricht genau 1000 Bytes!

In Bild 1 sehen Sie die acht Bytes, die in den Speicherstellen 53256 bis 53263 des Zeichen-ROM den Buchstaben A als Muster enthalten. Jedes Kreuz entspricht einem gesetzten Bit (also einem Bit mit dem Inhalt 1), die anderen enthalten 0. Das Betriebssystem liest in dem Moment, wo ein A auf dem Bildschirm erscheinen soll, dieses Zeichenmuster und legt es in den Bildschirmspeicher. Der Bildschirm ist nämlich ein RAM-Bereich und dort hinein kann ja geschrieben werden. Überall dort, wo nun ein gesetztes Bit im Bildschirmspeicher zu finden ist, leuchtet ein Bildpunkt auf. An allen anderen Stellen bleibt die Hintergrundfarbe erhalten. In welcher Farbe der Bildpunkt aufleuchten soll, erkennt unser Computer an einem weiteren RAM-Bereich, der genauso angeordnet ist wie der Bildschirm, nämlich dem Bildschirmfarbspeicher. Im Handbuch sind beide Speicherbereiche beschrieben und abgebildet. Nun wissen wir also, wie die normalen Zeichenmuster vom Computer benutzt werden. Weil diese sich im ROM befinden, kann man sie nicht verändern, denn dazu müßte man dort ja andere Muster hineinschreiben. Schreiben kann man aber nur in RAM-Bereiche. Deshalb kopiert man nun einfach den Inhalt des Zeichen-ROM in einen RAM-Bereich. Der VIC-II-Chip bietet außerdem die Möglichkeit, anzugeben, woher die Zeichenmuster geholt werden sollen. Damit steht uns im Prinzip der Weg zum Verändern der Muster offen. Wie findet nun das eigentliche Verändern statt? Dazu muß man wissen, daß jedem Byte eines solchen Zeichenmusters eine Zahl zugeordnet werden kann. Die berechnet man so, daß man jedem gesetzten Bit je nach seiner Bitnummer einen bestimmten Wert zuordnet. All diese Werte zählt man schließlich zusammen. Welche Werte zu welcher Bitnummer gehören, zeigt Ihnen Bild 2:

Ein Beispiel sehen Sie in Bild 3. Wie Sie beim Vergleichen mit Bild 1 erkennen werden, handelt es sich um den Inhalt der Speicherstelle 53256, das erste Byte des Zeichenmusters von A also. Eine Veränderung erreicht man nun durch Einschreiben eines anderen Wertes.

Wir werden nun Schritt für Schritt ein Rezept entwickeln, mit dem wir einem Zeichen, beispielsweise dem Buchstaben A, ein völlig neues Aussehen verleihen können. Das so entwickelte Programm ist natürlich jederzeit ausbaufähig für weitere Zeichenmusteränderungen.

Schritt 1: Wir packen den Inhalt des Zeichen-ROM ins RAM ab Speicherstelle 12288. Dies ist normalerweise ein Bereich, in dem lange Basic-Programme oder deren Variable, Felder oder Strings auftauchen können. Dadurch aber würde unser Zeichensatz überschrieben werden. Es gibt die Möglichkeit, den Basic-Speicher so einzuschränken, daß er nur noch bis 12287 reicht. Das machen wir mit:

```
40 POKE 52,48:POKE 56,48
```

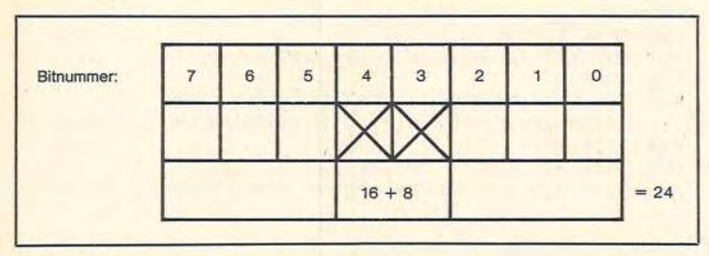


Bild 3. Ein Bitmuster wird zur Zahl

Schritt 2: Unser C 64 enthält ein ständig nebenher laufendes – von uns nicht bemerktes – Maschinenprogramm, das den normalen Betrieb möglich macht. Beispielsweise steuert dieses Programm das Blinken des Cursors, es sorgt für das Weiterlaufen der internen Uhr (TI$) und schaltet blitzschnell verschiedene ROM-Bausteine – auch das Zeichen-ROM – an oder aus. Wenn wir den Inhalt des Zeichen-ROM kopieren wollen, dürfen wir natürlich währendessen nicht unterbrochen werden. Es gibt eine Speicherstelle im sogenannten CIA 1, mit deren Inhalt man solche Unterbrechungen aus- und einschalten kann. Wir schalten diesen Störenfried ab:

```
60 POKE 56334,PEEK(56334) AND 254
```

Nebenbei bemerkt: Sie werden sich natürlich fragen, wie es denn zu all diesen Zahlenangaben kommt und was beispielsweise die Operation AND an dieser Stelle bedeutet. Im Verlauf dieses Artikels werden sich diese und weitere Fragen noch häufen. Wenn Sie – wie in diesem Schnellkurs vorgesehen – nicht nur nachkochen möchten, sondern auch verstehen, was hier geschieht, dann verweise ich Sie nochmal auf die eingangs genannte Literatur. Unser Artikel würde mit allen zu erklärenden Grundlagen über 100 Seiten Umfang annehmen.

Schritt 3: Weil wir nach dem Abschalten der Unterbrechung gar nicht wissen, ob das Zeichen-ROM gerade eingeschaltet war oder nicht, schalten wir es sicherheitshalber einfach ein. Der sogenannte Prozessorport in Speicherstelle 1 dient solchen Zwecken:

```
80 POKE 1,PEEK(1) AND 251
```

Schritt 4: Wir sind nun bereit zum Kopieren. Das Zeichen-ROM beginnt bei Adresse 53248. Wir kopieren 4096 Byte, was 512 Zeichenmustern entspricht. Die Tabelle 2 zeigt Ihnen die Anordnung der verschiedenen Zeichenmuster im ROM:

Als Zieladresse haben wir in Schritt 1 uns auf 12288 festgelegt. Eigentlich bräuchten wir nun nur zu programmieren:

```
Zeilennummer  FOR I=0 TO 4095:POKE 12288+I,
PEEK(53248+I):NEXT
```

Wenn wir nachher das gesamte Programm vorliegen haben, können Sie ja mal spaßeshalber anstelle der Zeilen 90 bis 180 diese Programmzeile verwenden. Es funktioniert! Aber das Kopieren damit dauert so lange, daß man in der Zwischenzeit gemütlich eine Tasse Tee trinken kann. Dasselbe – nur erheblich schneller – leistet ein Maschinenprogramm, das die beiden folgenden Zeilen ab Speicherstelle 49152 erzeugt:

```
100 FOR I=49152 TO 49158:READ A:POKE I,A:NEXT I
110 DATA 133,95,134,96,76,191,163
```

Bevor wir dann das Maschinenprogramm zum Kopieren aufrufen, müssen ihm noch die Adressen des Quell- und des Zielspeicherbereiches übergeben werden. Hier legen wir zunächst die Anfangs- und die Endadresse des Zeichen-ROM fest:

```
130 QS = 53248:QE = QS + 4096
```

Des weiteren verlangt das verwendete Maschinenprogramm noch die Endadresse des Zielspeicherbereiches:

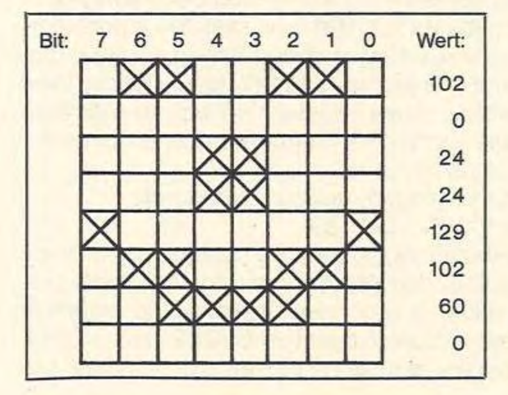


Bild 4. Das neue A

Block	Adressenbereich	Zeichen	Muster abrufbar im Programm mit Code
0	53248-53759	Satz 1 von 0 bis 63 (0 bis ?)	0- 63
	53760-54271	Satz 1 von 64 bis 127 (Grafikz.)	64-127
	54272-54783	Satz 1 von 0 bis 63 (Reversed)	128-191
	54784-55295	Satz 1 von 64 bis 127 (Reversed)	192-255
1	55296-55807	Satz 2 von 0 bis 63 (kleine Buchst.)	256-319
	55808-56319	Satz 2 von 64 bis 127 (Großbuchst.+Grafikz.)	320-383
	56320-56831	Satz 2 von 0 bis 63 (Reversed)	384-447
	56832-57343	Satz 2 von 64 bis 127 (Reversed)	448-511

Tabelle 2. Inhalt des Zeichen-ROM

```
140 ZE = 12288 + 4096
```

Die Adressenübergabe geschieht in einer bestimmten Form (im sogenannten LSB/MSB-Format) über festgelegte Speicherstellen:

```
150 A = INT(QS/256):POKE 781,A:POKE 780,QS-256*A
160 A = INT(QE/256):POKE 91,A:POKE 90,QE-256*A
170 A = INT(ZE/256):POKE 89,A:POKE 88,ZE-256*A
```

Nach all diesen Vorbereitungen rufen wir das Maschinenprogramm zum Kopieren nun auf:

```
180 SYS 49152
```

Schritt 5: Nach all diesen Eingriffen stellen wir den Normalzustand unseres Computers wieder her. Das Zeichen-ROM wird zuerst wieder abgeschaltet:

```
200 POKE 1,PEEK(1) OR 4
```

Dann erlauben wir wieder die Unterbrechungen:

```
210 POKE 56334,PEEK(56334) OR 1
```

Schritt 6: Noch liest unser Computer alle seine Zeichenmuster aus dem Zeichen-ROM. Das soll nun anders werden! Eine Speicherstelle im VIC-II-Chip steuert den Zugriff auf die Zeichenmusterquellen. Dort verändern wir nun den Inhalt derart, daß künftig diese Muster aus unserer Kopie ab 12288 gelesen werden:

```
230 POKE 53272,(PEEK(53272) AND 240) OR 12
```

Wenn Sie bis hierher das Programm abgetippt haben und es nun durch RUN starten, dann werden Sie überhaupt keinen Unterschied zum vorherigen Zustand feststellen. Das ist auch richtig so, denn wir haben ja noch kein Zeichen verändert. Das geschieht erst jetzt im nächsten Schritt.

Schritt 7: Wir legen zuerst einmal fest, welches Zeichen wir ändern möchten und bei welcher Speicherstelle sein Muster beginnt. Den Buchstaben A haben wir uns ausgesucht, zu dem der POKE-Code 1 gehört (diese Codes finden Sie im Handbuch):

```
250 C = 1
260 S = 12288 + 8*C
```

Als nächstes folgt eine kleine Schleife, die aus DATA-Zeilen die neuen Byte-Werte herausliest und sie dann in die zu A gehörigen Speicherstellen unseres Zeichen-RAM hineinschreibt:

```
270 FOR I=0 TO 7
280 :READ A:POKE S+I,A
290 NEXT
```

Spätestens jetzt sollten wir uns überlegen, wie unser neues A aussehen soll. Das geschieht im nächsten Schritt.

Schritt 8: Wir zeichnen uns dazu ein 8 mal 8 Raster, in das wir das neue Muster eintragen. Jede horizontale Reihe entspricht einem Byte, dessen Wert wir nun auf die oben beschriebene Weise (siehe Bilder 2 und 3) berechnen. Ein Beispiel zeigt Ihnen Bild 4:

Diese 8 Werte legen wir in eine DATA-Zeile:

```
300 DATA 102,0,24,24,129,102,60,0
```

Haben Sie alles abgetippt? Dann geben Sie nun einmal RUN ein und freuen sich über unser freundliches A. Als Programm »Abrakadabra« (Listing 2) finden Sie alles noch einmal zusammengefaßt und etwas ergänzt hier abgedruckt:

Die Ergänzungen bestehen aus der Zeile 30, in der der Bildschirm gelöscht und das Wort ABRAKADABRA gedruckt

wird, sowie aus dem Schlußteil. Das Programm meldet sich nämlich bei Zeile 320 mit einer BREAK-Mitteilung. Sie können nun mit dem neuen Zeichen herumexperimentieren (beispielsweise das ganze Programm mal LISTen). Falls Sie dann irgendwann CONT eingeben, wird auch noch die Zeile 330 durchlaufen, in der zunächst der VIC-II-Chip wieder auf das Zeichen-ROM gerichtet und dann auch die Begrenzung des Speichers rückgängig gemacht wird. Dieses unser zweites Rezept führt zu einem recht vielseitig verwendbaren Gericht. Brauchen Sie für irgendwelche Zwecke mal griechische Buchstaben? Dann nehmen Sie doch einfach den Bereich für die Reverse-Zeichen und erstellen darin griechische Zeichenmuster. Durch Einschalten des Reverse-Modus schreiben Sie dann griechisch. Benötigen Sie spezielle Grafikzeichen, die nicht auf der Tastatur vorgegeben sind? Sie wissen nun, wie Sie sie erzeugen und benutzen können. Größere grafische Objekte setzen Sie sich einfach aus mehreren verschiedenen veränderten Zeichen zusammen. Durch eine Anzahl von PRINT-Befehlen können Sie sie dann jederzeit erzeugen. Bleiben wir noch ein wenig bei den Zeichen im nächsten Rezept.

Zeichen im Mehrfarbenmodus

Erinnern wir uns: Auf dem Bildschirm wird ein Zeichen so abgebildet, wie es im Zeichenmuster gespeichert wurde. Überall dort, wo ein gesetztes Bit vorhanden ist, leuchtet ein Bildpunkt auf. Die Farbe des Bildpunktes entspricht dem Farbcode, der in der zur Bildschirmspeicherstelle gehörigen Bildschirmfarbspeicherstelle enthalten ist. Bild 5 verdeutlicht die Zusammenhänge:

Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel {CLR}, muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksummer 64 auf Seite 135.

```
10 REM *** NEUE ZEICHENMUSTER ***          <194>
20 PRINT CHR$(147)"ABRAKADABRA"            <004>
30 REM --- RAM SCHUETZEN ---               <193>
40 POKE 52,48:POKE 56,48                   <068>
50 REM --- UNTERBRECHUNG ABSCHALTEN ---    <030>
60 POKE 56334,PEEK(56334)AND 254           <215>
70 REM --- ZEICHENROM EINSCHALTEN ---      <177>
80 POKE 1,PEEK(1)AND 251                   <255>
90 REM --- ML-PROGRAMM ERZEUGEN ---        <093>
100 FOR I=49152 TO 49158:READ A:POKE I,A:N
    EXT I                                  <009>
110 DATA 133,95,134,96,76,191,163          <056>
120 REM --- KOPIEREN ZEICHENROM ---        <233>
130 QS=53248:QE=QS+4096:REM ZEICHENROM     <244>
140 ZE=12288+4096:REM NEUES ZEICHENRAM     <250>
150 A=INT(QS/256):POKE 781,A:POKE 780,QS-2
    56*A                                   <075>
160 A=INT(QE/256):POKE 91,A:POKE 90,QE-256
    *A                                     <159>
170 A=INT(ZE/256):POKE 89,A:POKE 88,ZE-256
    *A                                     <094>
180 SYS 49152                              <238>
190 REM --- NORMALZUSTAND HERSTELLEN ---   <036>
200 POKE 1,PEEK(1)OR 4                     <197>
210 POKE 56334,PEEK(56334)OR 1             <127>
220 REM -- VIC AUF NEUE ZEICHENQUELLE --   <244>
230 POKE 53272,(PEEK(53272)AND 240)OR 12   <071>
240 REM --- ZEICHEN A VERAENDERN ---       <224>
250 C=1:REM CODE FUER ZEICHEN A            <233>
260 S=12288+8*C:REM START MUSTER VON A     <199>
270 FOR I=0 TO 7                           <085>
280 :READ A:POKE S+I,A                     <029>
290 NEXT I                                 <120>
300 DATA 102,0,24,24,129,102,60,0          <210>
310 REM --- PROGRAMMENDE ---               <117>
320 STOP                                   <132>
330 POKE 53272,21:POKE 52,160:POKE 56,160:
    END                                    <190>
```

Listing 2. »Abrakadabra«

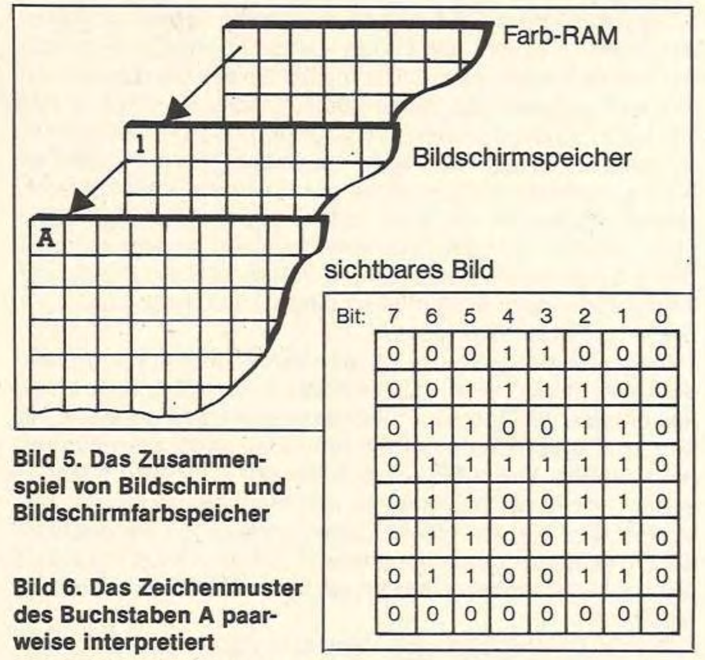


Bild 5. Das Zusammenspiel von Bildschirm und Bildschirmfarbspeicher

Bit: 7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
0	0	0	0	0	0	0	0

Bild 6. Das Zeichenmuster des Buchstaben A paarweise interpretiert

Die Farbe der Stellen, an denen kein gesetztes Bit vorliegt, stammt aus der Speicherstelle 53281. Das ist der Normalfall. Für Farbgourmets sehen wir uns nun ein Rezept an, das uns hilft, bunte Zeichen zu servieren. Es gibt nämlich einen Betriebszustand des VIC-II-Chip, in dem dieser die Zeichenmuster zur Farbgebung auf veränderte Weise liest. Hier entscheiden nicht mehr einzelne Bit-Inhalte, sondern immer Bit-Paare über die zu zeigende Farbe. Sehen wir uns das nochmal am Muster für den Buchstaben A an, das wir nun paarweise zusammenfassen (siehe Bild 6):

Es tauchen vier verschiedene Kombinationen auf: 00, 01, 10 und 11. Sie vermuten richtig, wenn Sie nun annehmen, daß wir vier verschiedene Farben in unseren Zeichen darstellen können. Die Kombination 00 entspricht dabei der 0 im normalen Modus: An dieser Stelle leuchtet kein Bildpunkt auf, die Farbe entstammt dem VIC-Register 53281 für die Hintergrundfarbe. An Stellen, an denen sich eine Kombination 01 befindet, leuchtet der Bildschirm auf. Als Farbinformation wird der Inhalt der Speicherstelle 53282 verwendet. Das ist ein VIC-Register, das Hintergrundfarbe 1 genannt wird. Auch an den Stellen mit der Kombination 10 leuchtet der Bildschirm auf. Die Farbinformation stammt hier aus dem VIC-Register 53283 (Hintergrundfarbe 2). Etwas komplizierter verhält sich das mit der Paarung 11. Hier wird jeweils die Farbe verwendet, die im dazugehörigen Bildschirmfarbspeicher (also in dem Speicher, der für die Farbe der Zeichen im Normalmodus eine Rolle spielt, ab 55296) angegeben ist. Allerdings findet hier nur ein Teil dieses Farbcodes Verwendung. Man kann zwar alle 16 möglichen Farbcodes (siehe Handbuch) eingeben, aber tatsächlich treten im Ergebnis nur die Codes 0 bis 7 auf: Die Codes werden als MODULO(8) gedeutet (MODULO(8) bedeutet, daß der Rest nach einer Division durch 8 verwendet wird). Auf diesen Code aber kommen wir gleich nochmal zurück. Bisher wissen Sie ja noch gar nicht, wie wir unseren Computer dazu bringen können, die Zeichenmuster auf diese extravagante Weise zu lesen. Des Rätsels Lösung liegt in einem VIC-II-Chip-Register, nämlich 53270. Man schaltet diesen Mehrfarbenmodus ein mittels:

```
POKE 53270,PEEK(53270) OR 16
```

In den Normalmodus zurück gelangt man durch:

```
POKE 53270,PEEK(53270) AND 239
```

Falls Sie das einfach mal ganz direkt ausprobieren, dann werden Sie feststellen, daß das Ergebnis (mehrfarbige Zeichen oder nicht) außer von diesem Register und natürlich dem Inhalt der beiden Speicherstellen 53282 und 53283 auch noch stark von der aktuellen Zeichenfarbe abhängt. Da

haben wir nämlich die zweite Voraussetzung für die Erzeugung von Mehrfarbenzeichen: Die aktuelle Zeichenfarbe muß einen Code aufweisen, der größer als 7 ist. Ist der Code (steuerbar durch POKE 646,Farbcode) kleiner oder gleich 7, dann ändert sich zwar die Farbe gemäß dem eingegebenen Code, aber weit und breit ist nichts vom Mehrfarbenmodus zu sehen. Sobald wir aber auf einen Code ab 8 kommen, wird es bunt. Wie das aussieht, können Sie nachvollziehen durch das Programm »Bunte Zeichen« (Listing 3):

Nach dem RUN erscheint zunächst die Schrift im Normalmodus. Das ist der Fall, weil hier der Farbcode der Zeichen noch kleiner als 8 ist, nämlich 1. Danach sehen Sie die bunten Zeichen. Der Farbcode wird nämlich in Zeile 90 auf 10 eingestellt. Mit den Farben ist das übrigens immer ein kleines Glücksspiel: Nahezu jeder Monitor (vom Farbfernseher ganz zu schweigen) erzeugt andere Farben. Um die für Ihr Gerät günstigste Kombination auszuprobieren, sollten Sie einfach mal in den Zeilen 30 und 90 andere Farbcodes ausprobieren. So! Nach all diesen Vorspeisen kommen nun die Hauptgerichte. Als erstes ein Rezept für Sprites:

Sprites

Sprites sind ungewöhnliche grafische Objekte. Sie kümmern sich überhaupt nicht darum, ob unser VIC-II-Chip gerade den Textmodus oder einen grafischen Modus verwaltet. Wie seltsam das ist, werden Sie später bemerken, wenn wir feststellen, daß die gleichzeitige Darstellung von Text und Grafik (wenn es keine Grafik aus den Tastaturzeichen ist) nur schwer möglich ist. In Bild 7 finden Sie einen Plan der Kochvorschrift, mit deren Hilfe jedes Sprite programmiert werden kann.

Diesem Schema werden wir nun Schritt für Schritt folgen bis zum fertigen Sprite-Programm. Der krönende Abschluß dieses Rezeptes wird dann ein kleines Programm sein, das Ihnen erklärt, weshalb der fliegende Holländer – der in früheren Jahrhunderten ständig die Weltmeere unsicher machte – seit vielen Jahrzehnten keine Schlagzeilen mehr in der Presse verursacht. Wo ist er geblieben? Dazu später.

Schritt 1: Grobplanung

Der erste Schritt erfordert etwas Planung: Wofür brauchen wir die Sprites? Ist es sinnvoll, einen einfarbigen – dafür aber feiner strukturierten – oder besser einen mehrfarbigen (drei Spritefarben + Hintergrundfarbe) zu benutzen, der allerdings dann weniger Details aufweist? Von dieser Entscheidung hängen einige weitere in späteren Schritten ab. Wir werden hier einfach zwei Sprites verwenden, von denen einer ein- und der andere mehrfarbig ist.

```
10 REM *** BUNTE ZEICHEN ***                      <132>
20 PRINT CHR$(147):POKE 53280,0                   <215>
30 POKE 53281,0:POKE 53282,7:POKE 53283,5:
   REM LADEN DER 3 FARBREGISTER                   <226>
40 POKE 53270,PEEK(53270)OR 16:REM ANSCHAL
   TEN DES MEHRFARBENMODUS                        <231>
50 POKE 646,1:REM ZEICHENFARBE AUF 1(WEISS
   )                                              <229>
60 PRINT CHR$(17)"JETZT IST DER MEHRFARBEN
   MODUS":PRINT"EINGESCHALTET"                    <136>
70 PRINT CHR$(17)"IN DIE BILDSCHIRMFARBZEI
   LEN IST ABER DIE";:PRINT"1 EINGEGEBEN"         <220>
80 PRINT CHR$(17)"DESWEGEN IST DIE ZEICHEN
   DARSTELLUNG":PRINT"NORMAL"                     <007>
90 POKE 646,10: REM ZEICHENFARBE AUF 10(HR
   OT)                                            <046>
100 PRINT CHR$(17)"DER FARBCODE IST JETZT
    10"                                           <144>
110 PRINT CHR$(17)"DER MEHRFARBEN-MODUS IS
    T SICHTBAR"                                   <089>
120 END                                           <122>
130 POKE 53270,PEEK(53270) AND 239:REM AUS
    SCHALTEN DES MEHRFARBEN-MODUS                 <132>
140 END                                           <142>
```

Listing 3. »Bunte Zeichen«

Sprite progr.
Einfarbiges Sprite ?
N (also Multicolor-Sprite)
J
Konstruktion des Sprite im Bit-Raster — a
Konstruktion des Sprite im Doppelbit-Raster — b
Übersetzen in Dezimalzahlen
Festlegen eines Speicherbereiches für Sprite-Daten
Einleseschleifen
Spritezeiger setzen — 2040... (Spritenummer festlegen) N
Sprite anschalten — 53269 (Bit N = 1)
Sprite-Ort festlegen X-Pos.: 53248 + N Y-Pos.: 53249 + N — dabei X-Pos. LSB MSB in 53264 (Bit N)
Einfarbiges Sprite ?
N
Sprite Farbcodes für 01 in 53285 11 in 53286 10 in 53287 + N
Sprite in Multicolor-modus umschalten Bit N von 53276 = 1 — b
J
Sprite-Farbcode in 53287 + N — a
Soll in X-Richtung verdoppelt werden ? — J — X-Verdoppeln Bit N in 53277 = 1
N
in Y-Richtung verdoppeln ? — J — Y-Verdoppeln Bit N in 53271 = 1
N
Haben Bildschirmdarstellungen Priorität ? — J — Prioritätenregister Bit N in 53275 = 1
N
Werden Zusammenstöße registriert ? — J — Kollisionsregister auslesen Sp/Sp 53278 Sp/Hg 53279
N
Fertig

Bild 7. Kochvorschrift fürs fertige Sprite

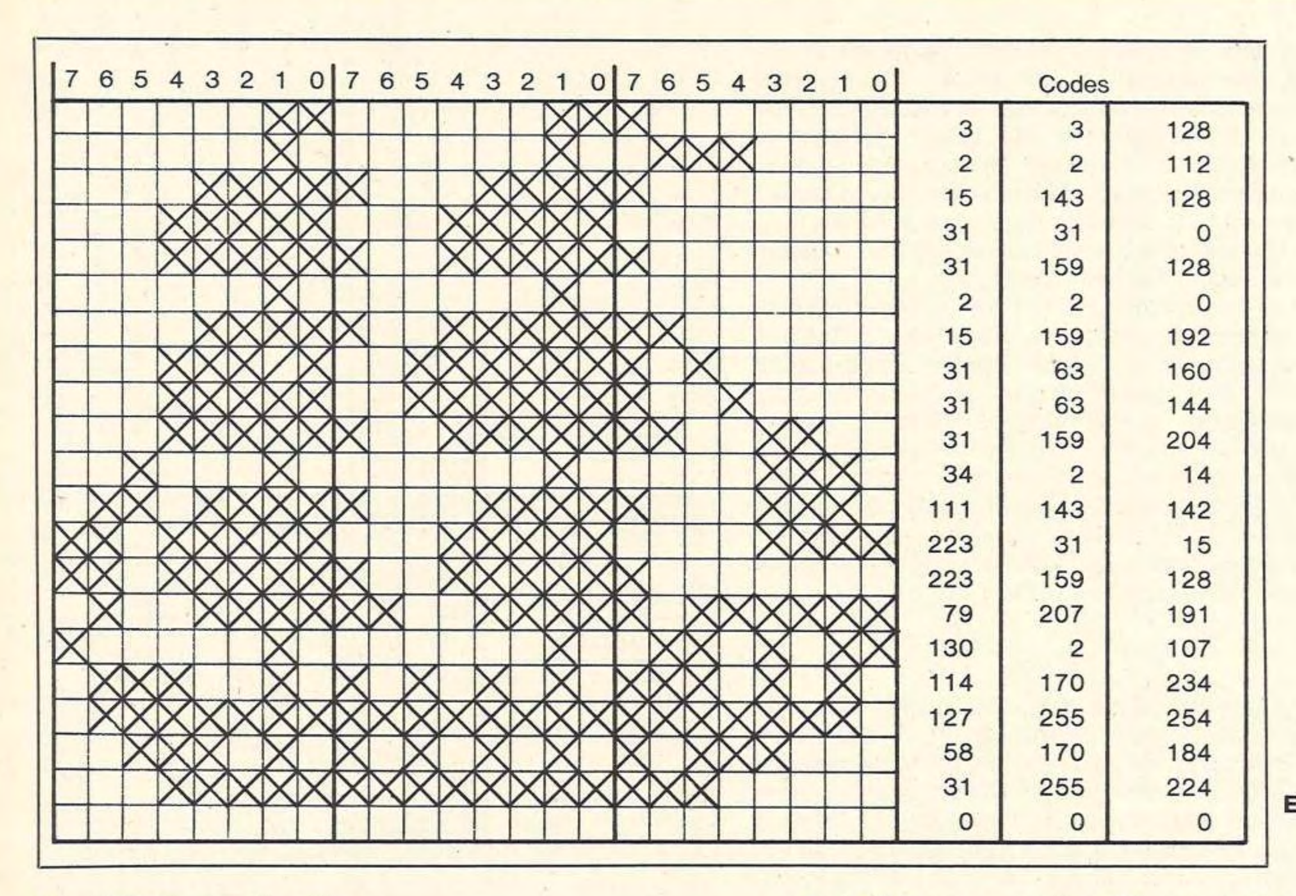


Bild 8. Entwurf eines einfarbigen Sprites

Schritt 2: Die kreative Phase

Wir spitzen den Bleistift und zeichnen uns auf kariertem Papier ein Rechteck mit 24 Kästchen waagerecht und 21 senkrecht. Das ist unser Sprite-Entwurfsblatt. Genau diese Ausdehnung – in Bildpunkten statt Kästchen – hat ein normales Sprite. Es folgt dann – hoffentlich – der beflügelnde Kuß der Musen. Hat Sie die Inspiration gepackt, dann kommt es darauf an, ob sie ein ein- oder ein mehrfarbiges Sprite kreieren.

a) Beim Einfarbigen plazieren Sie an jede Stelle, an der ein Bildpunkt leuchten soll, ein Kreuzchen: das sieht dann einem Stickmuster sehr ähnlich. Auf diese Weise ergibt sich auch unser erstes Sprite in Bild 8.

Etwas komplexer verhält sich das beim mehrfarbigen Sprite. Ähnlich wie bei den mehrfarbigen Zeichen, werden auch hier die Bits paarweise interpretiert. Dabei gibt es wieder vier verschiedene Kombinationsmöglichkeiten und daher auch vier Farben: 00 erzeugt die Hintergrundfarbe, 01, 10

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Codes		
1	1	1	1	1	1	1	1																	255	0	0
								1	1					1	1	1	1							0	195	192
										1	1	1	1					1	1					0	60	48
								1	1	1	1									1	1			0	240	12
										1	1											1	1	0	48	3
				1	1	1	1	1	1	1	1	1	1	1	1							1	1	15	255	3
		1	1							1	1			0	1	0	1	1	1	1	1			48	49	124
1	1	1	1							1	1	0	1	0	1	0	1	0	1					240	53	80
										1	1	0	1	0	1	0	1	0	1					0	53	80
										0	1	0	1	1	0	0	1	1	0	0	1			0	22	100
		1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1			63	213	84
1	1									0	1	0	1	0	1	0	1	0	1					192	21	80
								1	1	1	1	0	1	0	1	1	1	1	1					0	245	240
						1	1					1	1			1	1			1	1			3	12	204
						1	1			1	1					1	1					1	1	3	48	195
						1	1	1	1							1	1					1	1	3	192	195
						1	1							1	1							1	1	3	3	3
				1	1			1	1					1	1							1	1	12	195	3
1	1	1	1									1	1									1	1	240	12	3
										1	1					1	1			1	1			0	48	204
				1	1	1	1	1	1									1	1					15	192	48

Bild 9. Entwurf eines mehrfarbigen Sprites

und 11 führen jeweils zu verschiedenen Vordergrundfarben. Hier legt man sich am besten drei verschiedene Farbstifte bereit und füllt nun Bitpaar für Bitpaar mit der gewünschten Farbe aus. Zuvor ordnet man noch jeder Kombination eine Farbe zu. Unser Beispielssprite mit mehreren Farben ergibt sich auf diese Weise wie in Bild 9 gezeigt:

Schritt 3: Die Rechenphase

Damit ist die schöpferische Arbeit zunächst einmal beendet, und wir wenden uns profaner Mathematik zu. Ähnlich wie wir es schon im Rezept zu den eigenen Zeichenmustern getan haben, wandeln wir nun die Muster in einen Zahlencode um. Dazu unterteilen wir das Sprite-Entwurfsblatt durch zwei senkrechte Linien in drei Spalten zu je 8 Kästchen. Diese entsprechen jeweils drei Byte in einer Zeile. Jedem Kreuzchen in Bild 8 (oder jeder 1 in Bild 9) entspricht wieder ein bestimmter Zahlenwert. Zur Hilfe bedienen Sie sich bitte wieder des Bildes 2, wo diese Werte den Bitnummern zugeordnet sind. Alle Werte eines Bytes werden dann addiert (wie beispielsweise in Bild 3 gezeigt). So ergeben sich pro Zeile drei Codes und insgesamt 63 Codezahlen pro Sprite. In den Bildern 8 und 9 sind die sich ergebenden Codes rechts neben dem Entwurfsblatt gezeigt.

Schritt 4: Spritedatenspeicher aussuchen

Genauso wie unser Computer Zeichen nur dann darstellen kann, wenn sie als Muster (das sind ja die Codes) im Speicher zu finden sind, braucht er auch das Spritemuster im Speicher. Wo wir die 63 Byte eines Spritemusters plazieren, diese Überlegung ist nun unser nächster Schritt. Zwei Regeln gilt es zu beachten bei der Auswahl von Speicherplätzen zu diesem Zweck:

1) Der Spritespeicher muß im gleichen 16 KByte-Bereich liegen wie der Bildschirm. Das ist im Normalfall dann der Speicherraum von 0 bis 16383.

2) Die Startadresse der Spritedaten muß glatt durch 64 teilbar sein. Daneben sollte man natürlich darauf achten, daß man die Daten nicht in Speicherbereiche packt, die das Betriebssystem oder der Basic-Interpreter braucht. Kritisch sind da vor allem die ersten 1023 Speicherplätze. Gleichwohl bieten sich hier vier Orte an für vier Sprites:

704 - 767 1. Sprite
832 - 895 2. Sprite
896 - 959 3. Sprite
960 - 1023 4. Sprite

Die drei letzten davon gehören teilweise zum Kassettenpuffer. Falls Sie also während eines Spriteprogrammes auf die Kassette zugreifen müssen, dann ist es nötig, daß Sie danach die Spritedaten wieder in diesen Puffer zurückschreiben. Wenn Sie mehr als vier Sprites benötigen (habe ich Ihnen schon erzählt, daß Sie bis zu 8 davon gleichzeitig auf dem Bildschirm zeigen können?), dann läßt es sich wohl nicht umgehen, auch den Basic-Speicher dafür zu benutzen, der bei 2048 beginnt. Ich verwende in solchen Fällen meistens den Raum am oberen Ende des 16 KByte-Abschnittes, also knapp unter 16383. Eine andere - aber auch etwas umständlichere - Methode ist es, den Basic-Start höher zu legen. In dem Fall können dann die Spritemuster an den Bildschirmspeicher anschließen. Wir wollen uns aber in dieser kleinen Einführung nicht mit diesen Details belasten, sondern gehen einfach davon aus, daß wir mit nur zwei Sprites arbeiten, die wir nach 704 (für das einfarbige Sprite) und 832 (für das bunte Sprite) legen.

Schritt 5: Spritedaten einlesen

Nun beginnen wir zu programmieren. Wir legen die Spritedaten in DATA-Zeilen (und zwar jeweils 63 Zahlen plus eine 0, also 64 Zahlen). Im Programm SPRITES sind das dann die Zeilen ab 620. Außerdem brauchen wir zwei Einleseschleifen:

```
70 FOR I = 704 TO 767:READ A:POKE I,A:NEXT I
80 FOR I = 832 TO 895:READ A:POKE I,A:NEXT I
```

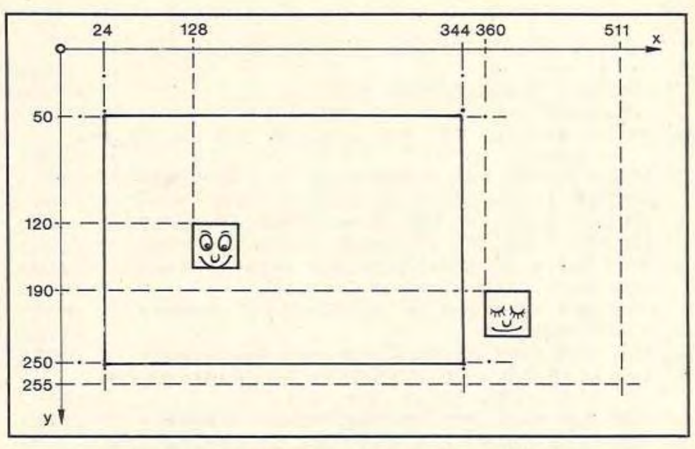
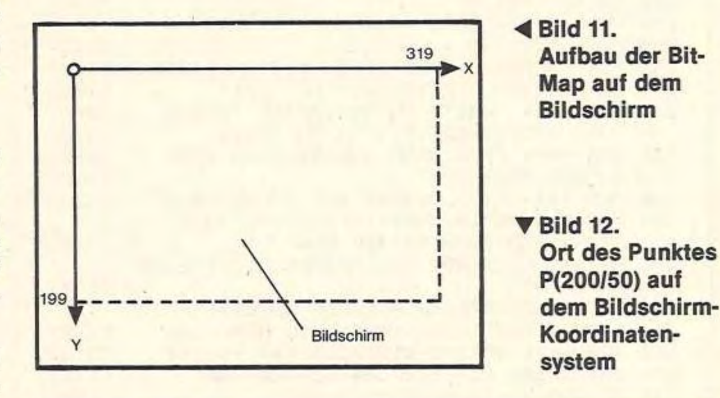
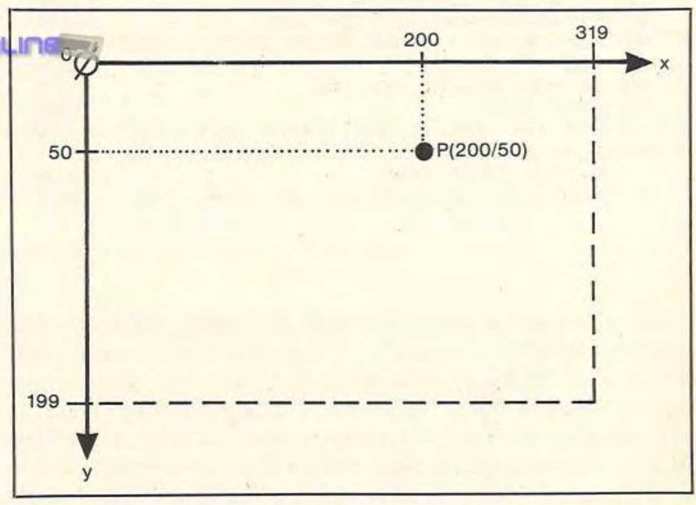


Bild 10. In diesem Feld können sich Sprites aufhalten. Dick umrandet ist der sichtbare Bildschirm.



◀ Bild 11. Aufbau der Bit-Map auf dem Bildschirm

▼ Bild 12. Ort des Punktes P(200/50) auf dem Bildschirm-Koordinatensystem



Damit hätten wir den aufwendigsten Teil der Sprite-Programmierung hinter uns.

Schritt 6: Wo liegen die Daten?

Dem Computer muß nun mitgeteilt werden, wo er die Spritemuster finden kann. Dazu gibt es direkt oberhalb des Bildschirmspeichers acht Adressen (ab 2040), in die eine Kennzahl der Spritedaten-Startadresse gepackt wird. Die Zuordnung zu einer solchen Speicherstelle legt gleichzeitig auch die Spritenummer fest. So zeigt die Kennzahl in 2040 automatisch immer auf das Sprite mit der Nummer 0, die in 2041 auf Sprite 1 und so weiter. Die Kennzahl ergibt sich aus der Startadresse des Spritemusters, dividiert durch 64:
Kennzahl = Startadresse / 64

Für unser einfarbiges Sprite wählen wir die Nummer 0 und schreiben nach 2040 nun 704/64 = 11. Für das andere Sprite wählen wir die Nummer 1 und für die Kennzahl gilt hier 832/64 = 13. Im folgenden soll immer N diese Spritenummer sein. Im Programm Sprites (Listing 4) findet diese Festlegung in Zeile 100 statt:

```
10 REM **** SPRITE - BEISPIELSPROGRAMM ***
   *                                        <214>
20 POKE 53280,0:POKE 53281,0               <148>
30 PRINT CHR$(147)CHR$(28)CHR$(17)CHR$(17) <050>
40 PRINT"DIES IST DIE LOESUNG DES RAETSELS
   ":PRINT                                  <038>
50 PRINT"UM DEN FLIEGENDEN HOLLAENDER"     <097>
60 REM ***** SPRITEDATEN EINLESEN *****    <181>
70 FOR I=704 TO 767:READ A:POKE I,A:NEXT I <152>
80 FOR I=832 TO 895:READ A:POKE I;A:NEXT I <128>
90 REM ***** SPRITE-ZEIGER ************    <168>
100 POKE 2040,11:POKE 2041,13              <093>
110 REM **** ZUM UP HINTERGRUND *******    <005>
120 GOSUB 550                              <176>
130 REM **** EINSCHALTEN **************    <182>
140 POKE 53269,PEEK(53269) OR (2↑0):POKE 5
    3269,PEEK(53269) OR (2↑1)              <179>
150 REM **** SPRITE-POSITIONEN ********    <096>
160 POKE 53248+2*0,250:POKE 53249+2*0,100:
    REM SPRITE 1                           <022>
170 POKE 53248+2*1,30:POKE 53249+2*1,190:R
    EM SPRITE 2                            <172>
180 POKE 53264,0                           <103>
190 REM **** SPRITE-FARBEN ************    <044>
200 POKE 53287,0                           <125>
210 POKE 53285,2:POKE 53286,5:POKE 53288,1 <167>
220 POKE 53276,PEEK(53276) OR (2↑1)        <180>
230 REM **** SPRITE-PRIORITAETEN ******    <066>
240 POKE 53275,PEEK(53275) OR (2↑1)        <198>
250 REM **** KOLLISION VORBEREITEN ****    <020>
260 A=PEEK(53278)                          <128>
270 REM **** DIE LOESUNG DES RAETSELS *    <122>
280 PRINT CHR$(19)CHR$(146)CHR$(5)"SEIT EI
    NIGEN JAHRZEHNTEN IST DER"             <164>
290 PRINT"FLIEGENDE HOLLAENDER NICHT MEHR
    AUF DEN"                               <014>
300 PRINT"WELTMEEREN GESEHEN WORDEN!":PRIN
    T CHR$(158)"HIER SEHEN SIE WESHALB:"   <128>
310 REM **** SPRITE-BEWEGUNG **********    <215>
320 D1=73/200:D2=-147/200:D3=-90/200       <111>
340 X1=250:X2=30:Y2=190:Y1=100             <129>
350 X1=X1+D2:X2=X2+D1:Y2=Y2+D3             <070>
360 POKE 53248,X1:POKE 53250,X2:POKE 53251
    ,Y2                                    <229>
370 IF PEEK(53278)THEN 400                 <162>
380 GOTO 350                               <150>
390 REM **** KOLLISION ! **************    <195>
400 POKE 53275,0:POKE 53281,7:FOR I=1 TO 1
    0:POKE 53281,6:NEXT I                  <160>
410 POKE 53269,PEEK(53269) AND (255-2↑0)   <072>
420 POKE 53250,X2:POKE 53251,Y1:FOR I=0 TO
     50:NEXT I                             <078>
430 REM **** ABGANG UND ENDE **********    <106>
440 POKE 53275,0                           <045>
450 X2=X2+D1:Y1=Y1-D3                      <123>
460 POKE 53250,X2:POKE 53251,Y1            <080>
470 IF Y1>251 THEN 490                     <091>
480 GOTO 450                               <002>
490 REM **** ENDE DER VORSTELLIUNG ****    <237>
500 POKE 53269,0:POKE 53281,0              <121>
510 PRINT CHR$(147)CHR$(30)CHR$(17)CHR$(17
    )                                      <156>
520 PRINT"VERSTAENDLICHERWEISE WARD SEITHE
    R"                                     <123>
530 PRINT"DER FLIEGENDE HOLLAENDER NIE MEH
    R"                                     <139>
535 PRINT"GESEHEN."                        <194>
540 END                                    <034>
550 REM *** UP HINTERGRUND ***********     <161>
560 PRINT CHR$(147)CHR$(17)CHR$(17)CHR$(17
    )CHR$(17)CHR$(17)CHR$(17)CHR$(17)      <174>
570 B$="UI":C$="KJ":FOR I=1 TO 20:A$=A$+B$
    :D$=D$+C$:NEXT I:POKE 53281,6          <129>
580 PRINT CHR$(154)CHR$(18);               <082>
590 FOR I=1 TO 8                           <056>
600 PRINT A$D$;:NEXT I                     <160>
610 RETURN                                 <160>
620 REM *** DIES SIND DIE SPRITEDATEN ***  <110>
630 REM ----- SPRITE 1 -----               <114>
640 DATA 003,003,128,002,002,112,015,143,1
    28,031,031,000,031,159,128,002,002     <232>
650 DATA 000,015,159,192,031,063,160,031,0
    63,144,031,159,204,034,002             <137>
660 DATA 014,111,143,142,223,031,015,223,1
    59,128,079,207,191,130,002,107,114     <117>
670 DATA 170,234,127,255,254,058,170,184,0
    31,255,224,000,000,000,000             <094>
680 REM ----- SPRITE 2 -----               <228>
690 DATA 255,000,000,000,195,192,000,060,0
    48,000,240,012,000,048,003,015,255     <150>
700 DATA 003,048,049,124,240,053,080,000,0
    53,080,000,022,100,063,213             <095>
710 DATA 084,192,021,080,000,245,240,003,0
    12,204,003,048,195,003,192,195,003     <218>
720 DATA 003,003,012,195,003,240,012,003,0
    00,048,204,015,192,048,000             <134>
```

Listing 4. »Sprite«

Für jedes Sprite, das in eine Kollision verwickelt wurde, ist dort dann ein Bit auf 1 gesetzt. Wenn also beispielsweise das Sprite 0 mit Bildschirmzeichen zusammenstößt, dann findet man in Register 53279 den Wert 1. Die Berechnung des zu erwartenden Wertes kann wieder mittels der Bitwerttabelle in Bild 2 geschehen. Sind mehrere Sprites in eine Kollision verwickelt, addiert man die Bitwerte. Das Auslesen dieser Register geschieht einfach durch PEEK. Allerdings ist dieses Auslesen so gründlich, daß ein weiteres PEEK dort nichts mehr finden würde, denn durch den PEEK-Befehl wird der Inhalt der beiden Kollisionsregister gelöscht. Braucht man diesen Inhalt aber noch – beispielsweise für eine Untersuchung, welche Sprites zusammengestoßen sind –, dann muß man den Registerinhalt beim Auslesen gleich in eine Variable packen. Man schreibt dann beispielsweise A = PEEK (53278) und kann dann A weiterverarbeiten. Im Programm SPRITES wird diese Radikalkur verwendet, um das fragliche Register (hier ist es 53278, weil wir an der Sprite/Sprite-Kollision interessiert sind) zu löschen. Zeile 260 dient dazu:

```
260 A = PEEK(53278)
```

Uns interessiert weiterhin gar nicht, welche Sprites zusammenstoßen – wir haben ja nur die beiden – Sprite 0 und 1. Deshalb untersuchen wir in Zeile 370 lediglich, ob im Register 53278 überhaupt ein Wert, der von Null verschieden ist, auftaucht:

```
370 IF PEEK(53278) THEN 400
```

Falls Ihnen diese Schreibweise etwas ungewöhnlich vorkommt, dann lassen Sie sich sagen, daß damit das gleiche erreicht wird wie durch eine Abfrage IF PEEK(53278) < > 0 THEN... . Ist dann ein Zusammenstoß erfolgt, passiert durch die Zeile 370 zweierlei: Erstens befindet sich im Register 53278 dann der Wert 3, der ja von Null verschieden ist und zur Verzweigung des Programmes nach Zeile 400 führt. Zum anderen aber wird durch das PEEKen dieses Register gleich wieder gelöscht und steht bereit für weitere Kollisionsabfragen. Sind Sie etwas abergläubisch und irritiert es Sie, daß wir genau 13 Schritte zu machen hatten, um alle Spritemöglichkeiten auszuschöpfen? Dann glauben Sie sicherlich auch an die Sage vom geheimnisvollen Fliegenden Holländer, der einige Jahrhunderte die Seeleute aller Nationen in Angst und Schrecken versetzte, weil mit seinem Auftauchen immer ein Unglück verbunden war. Eigentlich war ja der Fliegende Holländer dazu verdammt, bis in alle Ewigkeit ruhelos über die Weltmeere zu segeln. Merkwürdigerweise findet man aber schon seit einigen Jahrzehnten keinerlei Bericht mehr – obwohl mit unseren hervorragenden Kommunikationsmitteln jedermann schnell davon erfahren würde – über ein Auftauchen dieses Unglücksschiffes! Woran liegt das? Dieses Rätsel kann nun erstmalig unser Beispielprogramm »Sprites« (Listing 4) lösen: Ich will Ihnen noch nicht zuviel verraten. Die Kommentare im Programm und unsere 13 Schritte erklären das Programm ausreichend. Im Grunde genommen

finden sich nur noch ein paar Ergänzungen, die aus Texten, einem Programmteil für den Hintergrund, einer Reaktion auf den Zusammenstoß der beiden Sprites und einer Schleife zur Bewegung von Sprites bestehen. Bei der Bewegung werden in jedem Schleifendurchlauf die X- und die Y-Koordinaten der beiden Sprites um einen bestimmten Betrag verändert und die neuen Werte in die Register für die Sprite-Orte geschrieben. Viel Spaß bei dieser Lösung des Rätsels, die eines unserer Hauptgerichte ist. Sehen wir uns nun ein weiteres Rezept für ein Hauptgericht an.

Hochauflösende Grafik

Wir haben nun schon einige Male erlebt – und zwar beim veränderten Zeichensatz und auch bei den Sprites –, daß einzelne gesetzte Bits einen leuchtenden Bildpunkt auf dem Bildschirm erzeugen und gelöschte Bits die Hintergrundfarbe bestehen lassen. Wie allerdings der VIC-II-Chip unsere Erzeugnisse auf den Bildschirm bringt, nämlich als 8 mal 8-Bit-Feld auf einmal bei den Zeichen oder als 24 mal 21-Bit-Feld bei den Sprites, darauf haben wir bisher nur geringen Einfluß ausüben können. Dabei braucht man nur einen kleinen Gedankensprung weiter zu gehen: Was wäre, wenn wir anstelle der eben erwähnten kompletten Bitmuster (8 mal 8 oder 24 mal 21) dafür sorgen könnten, daß nur einzelne Bits des Bildschirmes an- oder ausgeschaltet werden? Dann hätten wir 320 Bildschirmpositionen (nämlich 40 Spalten zu je 8 Bits) in der Waagerechten und 200 (das sind 25 Zeilen mit je 8 Positionen) in der Senkrechten zur Verfügung, könnten also insgesamt 64000 verschiedenen Bildpunkte steuern! Tatsächlich gibt es einen Betriebszustand des VIC-II-Chip, den sogenannten Bit-Map-Modus, der diese Steuerung erlaubt. Es ist wahrhaft ein Jammer, daß das Basic des C64 die darin liegenden Möglichkeiten nicht im geringsten unterstützt. Wieder müssen wir uns in ein Gewirr von POKE- und PEEK-Befehlen stürzen, wobei im Bit-Map-Modus auch noch eine große Portion Geduld vom Benutzer erwartet wird, solange er in der Sprache Basic arbeitet. Allerdings lohnt das Ergebnis die Geduld. »Map« kann man ins Deutsche als »Landkarte« übersetzen. Allerdings hinkt diese Entsprechung etwas, denn eine Landkarte versucht die Gegend auf dem Papier abzubilden, hier bei der Bit-Map aber ist es umgekehrt: Der Inhalt der Bit-Map wird auf dem Bildschirm gezeigt. Es handelt sich um einen 8000 Byte großen Speicherbereich, der für jeden Bildschirmpunkt von den 64000 möglichen ein Bit enthält (8000 Byte zu je 8 Bits = 64000 Bits). Ist dieses Bit auf 1 gesetzt, dann bildet es der VIC-II-Chip an einer bestimmten Position des Bildschirmes als leuchtenden Punkt ab. Die Aufgabe, Grafik zu programmieren, besteht also darin, innerhalb dieses Speicherbereiches die richtigen Bits auf 1 oder 0 zu setzen. Dazu kommen wir gleich noch. Nun also zum Rezept: Damit Sie die einzelnen Schritte auch gleich in der Anwendung sehen, ist hierzu noch ein kleines Programm »HiRes-Grafik« (Listing 5) abgedruckt:

```
10 REM **** HIRES - GRAFIK ****                <127>
20 POKE 53280,0:POKE 53281,0                   <148>
30 DEF FN A(X) = 50*SIN(X/30)+100              <070>
40 REM --- BASIC BEGRENZEN ---                 <055>
50 POKE 52,32:POKE 56,32                       <104>
60 REM --- FARBABFRAGE -------                 <045>
70 PRINT CHR$(147)CHR$(30)"HINTERGRUNDFARB
   E";:INPUT HF                                <221>
80 PRINT"VORDERGRUNDFARBE";:INPUT ZF           <140>
90 F = 16*ZF + HF                              <039>
100 PRINT CHR$(147)                            <129>
110 REM --- BITMAPMODUS EINSCHALTEN ---        <173>
120 POKE 53265,PEEK(53265) OR 32               <160>
130 REM --- BITMAPSTART AUF 8192 ------        <009>
140 POKE 53272,PEEK(53272) OR 8                <181>
150 REM --- BITMAP LOESCHEN (GEDULD!) -        <121>
160 B = 8192                                   <078>
170 FOR I=0 TO 7999                            <126>
180 :POKE B+I,0                                <105>
190 NEXT I                                     <018>
200 REM --- FARBGEBUNG (NOCHMAL GEDULD!)       <170>
210 C = 1024                                   <051>
220 FOR I=0 TO 999                             <205>
230 :POKE C+I,F                                <193>
240 NEXT I                                     <068>
250 REM --- PUNKTE ZEICHNEN -----------        <158>
260 FOR X=0 TO 319                             <100>
270 :Y = FN A(X)                               <175>
280 :BY = (X AND 504) + 40*(Y AND 248) + (
    Y AND 7)                                   <088>
290 :BI = 7 - (X AND 7)                        <139>
300 :POKE B+BY,PEEK(B+BY) OR (2↑BI)            <154>
310 NEXT X                                     <004>
320 REM --- WARTEN AUF TASTE ----------        <164>
330 GET A$:IF A$="" THEN 330                   <205>
340 REM --- NORMALMODUS EINSCHALTEN ---        <121>
350 POKE 53272,PEEK(53272) AND (255-8)         <000>
360 POKE 53265,PEEK(53265) AND (255-32)        <100>
370 PRINT CHR$(147)                            <145>
380 END                                        <128>
```

Listing 5. »HiRes-Grafik«

Schritt 1: Bit-Map-Modus einschalten

Das Register 53265 (genauer gesagt, das Bit 5 davon) ist zuständig für diesen Bit-Map-Modus. Man schaltet ihn ein durch:

```
POKE 53265,PEEK(53265) OR 32
```

und in den normalen Textmodus gelangt man durch:

```
POKE 53265,PEEK(53265) AND (255-32)
```

In unserem Programm »HiRes-Grafik« geschieht dieses Einschalten in Zeile 120. Probehalber können Sie den Einschaltbefehl auch mal im Direktmodus eingeben. Der sich dadurch ergebende Bildschirm ist recht interessant: Man kann nämlich dem Computer direkt bei der Arbeit zusehen, weil wir auf dem Bildschirm den Speicherbereich von 0 bis 999 abgebildet sehen. Dort befindet sich beispielsweise die sogenannte Zeropage, ein Arbeitsspeicher des Betriebssystems, in dem sich dauernd Inhalte verändern.

Schritt 2: Welche Bit-Map?

Stellen Sie sich vor, die auf die eben erwähnte Weise gewählte Bit-Map soll nun beschrieben werden. Das könnte ja dazu führen, daß Inhalte der Zeropage verändert werden müssen. Unser Computer ist in dieser Angelegenheit aber sehr eigen: Wenn man nicht sehr darauf achtet, wohin man was schreibt in diesem empfindlichen Speicherbereich, dann stürzt er ab. Es muß also noch eine Möglichkeit geben, einen anderen Ort für die Bit-Map anzugeben. Tatsächlich ist das der Fall und auf diese Weise packen wir die Bit-Map in einen Speicherbereich, in dem sie auch in den meisten Grafikprogrammen zu finden ist:

```
POKE 53272,PEEK(53272) OR 8
```

legt unsere Bit-Map-Startadresse nach 8192. Übrigens gibt es – wie Sie sich sicher denken können – noch mehrere Möglichkeiten, im Speicher unseres Computers solche Bit-Maps unterzubringen. Allerdings wird das etwas komplizierter als das hier gezeigte Verfahren, weshalb ich Sie da wieder auf die eingangs erwähnte Literatur verweisen möchte. Zeile 140 dient im Programm »HiRes-Grafik« zur Festlegung des Bit-Map-Ortes. Bevor wir uns den weiteren Programmverlauf ansehen – und damit die nächsten Schritte – soll noch der Aufbau der Bit-Map, wie er für uns auf dem Bildschirm existiert, anhand des Bildes 11 erklärt werden: Unser Bildschirm ist im Bit-Map-Modus praktisch wie ein Koordinatensystem aufgebaut, dessen Nullpunkt oben links liegt und dessen X-Werte von 0 bis 319 reichen, wohingegen die Y-Werte von oben (ab Null) nach unten verlaufen bis 199. Das ist ein etwas ungewöhnliches Koordinatensystem, mit dem sich aber auch allerhand anstellen läßt.

Schritt 3: Bit-Map schützen

Die mit diesen beiden POKE-Befehlen in die VIC-II-Chip-Register erzeugte Bit-Map liegt mitten im Basic-

Speicherraum (der beginnt ja bei 2048 und reicht bis 40960). Ähnlich wie bei der Erzeugung eigener Zeichen besteht auch hier die Gefahr, daß ein Basic-Programmtext oder Variable über 8192 hinausreichen. Dadurch würde aber die Bit-Map beeinflußt. Das kann man verhindern, indem man mittels zweier POKE-Befehle den Basic-Speicher begrenzt:

```
POKE 52,32:POKE 56,32
```

Damit hört der Basic-Speicher bei 8192 auf. Soll er wieder die normale Länge annehmen, dann gibt man ein:

```
POKE 52,160:POKE 56,160
```

Unser Programm vollführt diesen Schutz der Bit-Map in Zeile 50. Wenn Sie alles bisher vorgestellte eingegeben und durch RUN gestartet haben, dann sieht der Bildschirm sehr merkwürdig aus. Es zeigen sich mehr oder weniger zufällige Speicherinhalte als Punkte.

Schritt 4: Bit-Map säubern

Von diesen zufällig dort vorhandenen Inhalten müssen wir die Bit-Map natürlich reinigen, bevor wir sie ernsthaft für Zeichnungen benutzen. Das geschieht einfach durch eine Programmschleife, in der in jede Speicherzelle der Bit-Map eine Null geschrieben wird:

```
160 B = 8192
170 FOR I=0 TO 7999
180 :POKE B+I,0
190 NEXT
```

So sieht das dann auch im Beispielsprogramm »HiRes-Grafik« aus. Allerdings muß ich Sie noch warnen: das dauert eine ganze Weile! Hinterher ist der Bildschirm schwarz – jedenfalls im Programm. Wenn Sie alles bis hierher eingegeben und dann durch RUN gestartet haben, werden Sie meistens noch bunte Felder auf dem Bildschirm finden. Das hängt mit der besonderen Art der Farbgebung im Bit-Map-Modus zusammen, die wir uns nun ansehen werden.

Schritt 5: Es wird farbig

Anders als im Textmodus holt der VIC-II-Chip die Vorder- und die Hintergrundfarbe im Bit-Map-Modus aus dem Speicher, der bisher als Bildschirmspeicher gedient hat: aus dem Bereich 1024 bis 2023 also. Das gilt übrigens nur dann, wenn man den Bildschirm an der alten Stelle beläßt, so wie wir das hier tun. Es gibt aber auch Möglichkeiten, den Bildschirmspeicher zu verschieben. In dem Fall dient immer der aktuelle Bildschirmspeicher der Farbgebung. Das aber nur nebenbei. Jeder Bildschirmspeicherplatz (also beispielsweise die Speicherstelle 1024) ist zuständig für die Vorder- und die Hintergrundfarbe eines 8 mal 8-Bit-Bereiches der Abbildung (die eben erwähnte Speicherstelle liefert also die Farben für das Bildschirmquadrat, dessen linke obere Ecke die Koordinaten 0,0 und dessen rechte untere die Koordinaten 7,7 hat). Damit zwei Farbinformationen in einem Byte des Bildschirmspeichers Platz haben, bedient sich der Computer des sogenannten »gepackten« Formates. Das ist für uns einfach eine Kennzahl F, die sich aus der Vordergrundfarbe ZF und der Hintergrundfarbe HF durch folgende Formel berechnet:

$F = 16*ZF + HF$

Der Programmteil zwischen Zeile 60 und 90 fragt Sie nach der gewünschten Farbkombination und berechnet die Kennzahl F. Wir belegen nun den gesamten Bildschirmspeicher in einer Schleife mit dieser Kennzahl. Dadurch werden auch die eventuell vorhandenen bunten Felder – die sich aus zufälligen Inhalten des Bildschirmspeichers ergeben – überdeckt durch die gewählte Hintergrundfarbe:

```
210 C = 1024
220 FOR I=0 TO 999
230 :POKE C+I,F
240 NEXT
```

Auch dieser Teil unseres Programmes verlangt nochmal etwas Geduld von Ihnen. Damit hätten wir alle Vorbereitungen abgeschlossen. Falls Sie bis hierher alles abgetippt und mittels RUN gestartet haben, dann sehen Sie nun einen leeren sauberen Bildschirm vor sich, der in der Hintergrundfarbe erstrahlt. Nun können wir Punkte setzen. Aber wie macht man das?

Schritt 6: Punkte setzen

Einen Punkt auf dem Bildschirm sichtbar zu machen, erfordert – das hatten wir schon zu Beginn dieses Rezeptes erwähnt – das Setzen des richtigen Bits im richtigen Byte. Wir müssen also in den Bit-Map-Speicher etwas hineinschreiben. Probieren Sie doch einfach mal:

```
POKE 9000,255
```

Sie sehen plötzlich irgendwo im oberen Teil des Bildschirmes einen kleinen horizontalen Strich. Aber das ist es ja eigentlich nicht das, was wir brauchen. Wir wollen vielmehr ganz gezielt Punkte setzen können. Was tut man, um beispielsweise genau den Punkt anzuschalten, der einen X-Wert von 200 und einen Y-Wert von 50 sein eigen nennt (siehe Bild 12)? Das ist leider wegen des ziemlich komplizierten Speicheraufbaues der Bit-Map ein relativ komplizierter Weg. Wieder muß ich Sie da auf die Literatur verweisen. Wir brauchen uns hier nur einige Formeln zu merken. X und Y sind die Koordinaten des zu setzenden Punktes, BY ist ein Zwischenwert (der die Nummer des Bytes in der Bit-Map bezeichnet) und BI ebenfalls ein Zwischenwert, der die Nummer des Bit im Byte enthält. Die letzte Zeile sorgt dann für das Setzen des richtigen Bits in der Bit-Map:

```
280 BY = (X AND 504) + 40*(Y AND 248) + (Y AND 7)
290 BI = 7 - (X AND 7)
300 POKE B + BY,PEEK(B+BY) OR (2↑BI)
```

B ist übrigens im Programm (daraus stammen diese Zeilen) vorher schon auf 8192 festgelegt worden. Diese drei Zeilen lassen sich auch gut als Unterprogramm einsetzen.

Das war's eigentlich schon. Alles weitere im Programm »HiRes-Grafik« dient dazu, festzulegen, welche Punkte zu setzen sind.

Die Programmzeilen 260, 270 und 310 steuern eine Schleife, in der X von 0 bis 319 läuft.

In Zeile 270 wird mittels der Funktion A(X) der aktuelle Y-Wert berechnet. Das Programm zeichnet eine Sinusfunktion. Die Funktionsdefinition in Zeile 30 sieht allerdings etwas merkwürdig aus. Das rührt daher, daß wir solch ein ungewöhnliches Koordinatensystem zur Verfügung haben. Wenn Sie das Programm gestartet haben, werden Sie aufgefordert, die Vorder- und die Hintergrundfarbe einzugeben. Danach wird einige Geduld von Ihnen erwartet, bis schließlich die Sinuskurve auf dem Bildschirm erscheint. Das Programm wartet nun auf einen Tastendruck, um dann den normalen Textmodus wieder herzustellen. Dies also war wieder ein Hauptgericht. Kommen wir nun noch zur Nachspeise.

Es kommt natürlich immer darauf an, was Sie zeichnen wollen: Für die grafische Darstellung mathematischer Funktio-

BIT-PAAR	FARBQUELLE
00	SPEICHER 53281, HINTERGRUNDREG. Nr. 0
01	BITS 4-7 } DES VIDEO-RAM
10	BITS 0-3 } DES VIDEO-RAM
11	BILDSCHIRMFARBSPEICHER

Bild 13. Bitpaare und Farbquellen

```
10 REM **** MULTIGRAFIK ****                          <040>
20 POKE 53280,0:POKE 53281,0                          <148>
30 DEF FN A(X) = 40*SIN(X/5)+100                      <086>
32 DEF FN B(X) = 45*SIN(X/10)+100                     <160>
34 DEF FN C(X) = 50*SIN(X/15)+100                     <114>
40 REM --- BASIC BEGRENZEN ---                        <055>
50 POKE 52,32:POKE 56,32                              <104>
60 REM --- FARBABFRAGE -------                        <045>
70 PRINT CHR$(147)CHR$(30)"HINTERGRUNDFARB
   E";:INPUT F0                                       <108>
80 PRINT"VORDERGRUND 01{2SPACE}";:INPUT F1 <072>
82 PRINT"VORDERGRUND 10{2SPACE}";:INPUT F2 <009>
84 PRINT"VORDERGRUND 11{2SPACE}";:INPUT F3 <076>
90 F = 16*F1 + F2                                     <095>
100 PRINT CHR$(147)                                   <129>
110 REM --- BITMAPMODUS EINSCHALTEN ---               <173>
120 POKE 53265,PEEK(53265) OR 32                      <160>
130 REM --- BITMAPSTART AUF 8192 ------               <009>
140 POKE 53272,PEEK(53272) OR 8                       <181>
142 REM --- MEHRFARBENMODUS EIN -------               <247>
144 POKE 53270,PEEK(53270) OR 16                      <076>
150 REM --- BITMAP LOESCHEN (GEDULD!) -               <121>
160 B = 8192                                          <078>
170 FOR I=0 TO 7999                                   <126>
180 :POKE B+I,0                                       <105>
190 NEXT I                                            <018>
200 REM --- FARBGEBUNG (NOCHMAL GEDULD!)              <170>
210 C = 1024:D=55296                                  <106>
220 FOR I=0 TO 999                                    <205>
230 :POKE C+I,F                                       <193>
232 :POKE D+I,F3                                      <057>
240 NEXT I                                            <068>
242 POKE 53281,F0                                     <144>
250 REM --- PUNKTE ZEICHNEN -----------               <158>
255 PF=1                                              <195>
260 FOR X=0 TO 159                                    <103>
270 :Y = FN A(X)                                      <175>
280 :GOSUB 400                                        <195>
290 NEXT X                                            <240>
295 PF=2                                              <013>
300 FOR X=0 TO 159                                    <143>
302 :Y = FN B(X)                                      <015>
304 :GOSUB 400                                        <219>
310 NEXT X                                            <004>
312 PF=3                                              <062>
314 FOR X=0 TO 159                                    <157>
316 :Y = FN C(X)                                      <093>
318 :GOSUB 400                                        <233>
319 NEXT X                                            <013>
320 REM --- WARTEN AUF TASTE ----------               <164>
330 GET A$:IF A$="" THEN 330                          <205>
340 REM --- NORMALMODUS EINSCHALTEN ---               <121>
342 POKE 53270,PEEK(53270) AND (255-16)               <209>
350 POKE 53272,PEEK(53272) AND (255-8)                <000>
360 POKE 53265,PEEK(53265) AND (255-32)               <100>
370 PRINT CHR$(147)                                   <145>
380 END                                               <128>
400 REM --- UP PUNKTE SETZEN ----------               <031>
410 HX=8*INT(X/4)                                     <119>
420 HY=320*INT(Y/8)+(Y AND 7)                         <170>
430 BX=4↑(3-(X AND 3))                                <021>
440 BM=PF*BX                                          <059>
450 H1=B+HX+HY                                        <071>
460 POKE H1,(PEEK(H1) AND (NOT 3*BX)) OR B
    M                                                 <176>
470 RETURN                                            <018>
```

Listing 6. »Multigrafik«

nen ist es durchaus vorteilhaft, nur eine Zeichenfarbe zu verwenden. Für künstlerische Zwecke allerdings wären mehrere Farben erwünscht. Wie schon bei den Zeichen und den Sprites, gibt es natürlich auch im Bit-Map-Modus die Möglichkeit, vier Farben auf dem Bildschirm zu zeigen. Wieder werden nicht mehr einzelne Bits, sondern Bitpaare vom VIC-II-Chip interpretiert. Einen gravierenden Nachteil hat dieser Mehrfarben-Bit-Map-Modus allerdings:

In der Horizontalen stehen uns nun nicht mehr 320, sondern nur noch 160 Bildschirmpositionen zur Verfügung. Das schränkt die Anwendung dieses Modus gewaltig ein. Sehen wir uns die Zuordnung der Bitpaare zu den Farbquellen an (siehe Bild 13): Die Kombination 00 steht wieder für die Hintergrundfarbe. Der VIC-II-Chip holt sich den dazugehörigen Farbcode aus der Speicherstelle 53281. 01 und 10 vereinigen sich, wie zuvor im normalen hochauflösenden Modus die Vorder- und die Hintergrundfarbe, zu einem gepackten Format. Das befindet sich dann im Bildschirmspeicher. Ist F1 der Code für die Farbe, die zur Bitpaarung 01 gehört und F2 der für das Paar 10, dann berechnet sich die Kennzahl für beide aus der Formel:

F = 16*F1 + F2

Die Farbe der Bitkombination 11 schließlich stammt aus dem auch im Textmodus verwendeten Bildschirmfarbspeicher (55296 bis 56295). Drei Bereiche sind es, die im Vergleich zum normalen Bit-Map-Modus hier verändert werden müssen: Zum einen muß dem VIC-II-Chip gesagt werden, daß er nun die Bit-Map in Bitpaaren interpretieren soll. Zum zweiten müssen wir für die Belegung der verschiedenen Farbquellen Sorge tragen, und zum dritten verändert sich die Routine, die für uns aus den Koordinaten eines Punktes die richtige Stelle in der Bit-Map berechnet und dann dort Bits an- oder abschaltet. Das beigefügte Programm »Multigrafik« (Listing 6) soll Ihnen diese Unterschiede zeigen: Einschalten des Mehrfarben-Bit-Map-Modus: Daß er die Bits aus der Bit-Map nun paarweise interpretieren soll, erfährt der VIC-II-Chip durch das Register 53270. Dort schaltet man diesen Mehrfarben-Bit-Map-Modus ein durch:

```
144 POKE 53270,PEEK(53270) OR 16
```

Das Ausschalten geschieht im Programm in Zeile 342:

```
342 POKE 53270,PEEK(53270) AND (255-16)
```

Belegen der Farbregister: In den Programmzeilen 60 bis 90 ist die Abfrage auf die gewünschten Farben ergänzt worden ebenso wie die Berechnung der Farbkennzahl F. Die Farbgebung zwischen den Zeilen 200 und 242 wurde erweitert. Dazu wird in Zeile 210 der Variablen D die Startadresse 55296 des Bildschirmfarbspeichers übergeben. Die anschließende Schleife belegt dann nicht nur den Bildschirm-, sondern auch den Bildschirmfarbspeicher, letzteren mit der Farbe F3. In 242 wandert noch die Hintergrundfarbe F0 ins Register 53281.

Punkte zeichnen: Aus der einen Schleife, in der X von 0 bis 319 lief und die auch gleich die Berechnung der Bitadresse und den POKE-Befehl enthielt, wurden nun drei Schleifen, die jeweils eine etwas veränderte Sinusfunktion mit anderer Farbe auf den Bildschirm zaubern. Vor der Schleife erhält jeweils die Variable PF die gewünschte Farbkennzahl (sie ist hier übrigens identisch mit dem Dezimalwert des entsprechenden Bitpaares, also 11 = 3, das aber nur am Rande bemerkt). Das Berechnen und Zeichnen ist nun in ein Unterprogramm verschoben, welches bei 400 beginnt. Die Formeln in diesem Modus sind natürlich anders als im normalen hochauflösenden Modus. Wer darüber mehr wissen möchte, der sollte sich dazu das Commodore-64-Buch (Band 3) von Schneider und Eberl zu Gemüte führen. Es erscheint im Markt&Technik Verlag unter der Bestellnummer MT595. Das Unterprogramm erwartet beim Einsprung jeweils einen Koordinatenwert f=r X (0 bis 159), für Y (0 bis 199) und eine Farbzahl PF (0 bis 3). Ansonsten funktioniert »Multigrafik« genauso wie »HiRes-Grafik: Nach den Abfragen auf die gewünschten Farben dauert alles noch etwas länger, bis dann nacheinander die drei Kurven erscheinen. Wieder nach einem Tastendruck schaltet der VIC-II-Chip in den normalen Textmodus zurück.

Damit ist unsere Rezeptsammlung abgeschlossen. Natürlich bietet sie lediglich eine grundlegende Einführung in die Grafik-Kochkunst. Falls Ihnen das so gekochte Menü geschmeckt hat, dann wagen Sie sich doch mal an die raffinierteren Gerichte. Da gibt es beispielsweise im 64'er-Magazin einen Kurs »Grafik-Streifzüge« und natürlich die anfangs aufgezählte Literatur.

(Heimo Ponnath/cg)

Der leichte Umgang mit Sprites

Über Sprites wird immer wieder geredet, wenn es um den C 64 geht. Wie werden sie erstellt, auf den Bildschirm gebracht und bewegt? Hier soll nun alles über die »legendären« Sprites gesagt werden.

Wie bringt man Sprites auf den Bildschirm, wie werden sie gesteuert oder durch ein Programm beeinflußt? Hier erfahren Sie die Grundlagen, mit denen Sie Sprites erstellen und steuern können.

Ein Sprite ist ein frei definierbares, eigenständiges Zeichen, das auf dem Bildschirm dargestellt, bewegt und verändert werden kann, unabhängig vom jeweiligen Hintergrund. Das bedeutet nichts anderes, als daß ein Sprite auf jedem Hintergrund (Text- oder Grafikbildschirm) sichtbar werden kann. Ebenso ist es möglich, das Sprite vor oder hinter den Bildschirmzeichen zu bewegen. In der einfarbigen Darstellung betragen die Ausmaße 24 * 21 Pixel (Bildschirmpunkte), in mehrfarbiger Darstellung (Multicolor) schrumpfen dieses auf 12 * 21 Pixel (schrumpfen deshalb, da bedingt durch die Farbdarstellung zwei Punkte je sichtbarem Punkt herhalten müssen). Wie lassen sich aber jetzt die Sprites erzeugen und bewegen? Dazu müssen wir etwas über den Baustein erfahren, durch den dies alles erst machbar wird:

Der VIC (Video-Interface-Controller)

Den Namen trägt dieser Baustein zu Recht. Er kümmert sich nicht nur darum, die 40 * 25 Zeichen auf dem Bildschirm darzustellen, er sorgt auch für die hochauflösende Grafik und für einige computerinterne Steuerungen. Zu guter Letzt bringt er die Sprites auf den Bildschirm.

Gesteuert wird er über die Register des Video-Bereiches. Diese sind nichts weiter als die Speicherplätze des VICs. Sie beginnen ab der Adresse 53248 (Register 0) und enden bei 53294 (Register 46). In Tabelle 1 sehen Sie eine Übersicht über alle Videochip-Register mit den jeweiligen Funktionen.

Sie werden jetzt wahrscheinlich etwas verwirrt sein und sich fragen, was Sie mit dieser Tabelle anfangen sollen? Deshalb nun einige Erklärungen:

Die Register sagen dem VIC, wo, wie und ob überhaupt Sprites aktiviert werden sollen. Indem verschiedene Werte in die jeweiligen Register geschrieben (gePOKEt) werden, kann der Videoprozessor in seiner Funktion gelenkt werden.

Zuerst müssen Sie wissen, was die einzelnen Register des Video-Chips in ihrer Funktion bedeuten. Gehen wir sie der Reihe nach durch:

Register 0 bis 15: Hier wird die Position der einzelnen Sprites auf dem Bildschirm festgelegt. Je ein Registerpaar (0/1, 2/3 ...) ist für ein Sprite zuständig. Für das Sprite Nr. 1 zum Beispiel sind die Register 0 und 1 zuständig. Register 0 stellt die X-Position, Register 1 die Y-Position auf dem Bildschirm dar. Da auf diese Weise ein Sprite nur 256 Punkte auf der X-Achse bewegt werden kann (der Bildschirm aber 320 Punkte breit ist), finden wir im Register 16 die Überläufe der X-Positionen.

Register 16: Hier steht nun, ob ein Sprite auf dem Bildschirm in der Position 0 bis 255 (»0«) oder 256 bis 320 (»1«) erscheinen soll.

Register 17 bis 20, 22, 24 und 34 bis 36: Diese Register haben nichts mit der Sprite-Steuerung zu tun. Deshalb werden wir auch nicht näher darauf eingehen.

Register 21: Mit diesem Register sagen wir dem Computer, welche der 8 Sprites auf dem Bildschirm erscheinen sollen.

Register 23: Dient zum Vergrößern (Verdoppeln) der einzelnen Sprites in Y-Richtung.

Register 25 und 26: In diesen Registern wird angezeigt, ob ein Sprite ein anderes Sprite oder ein Hintergrundzeichen berührt.

Register 27: Hier können wir festlegen, welches Sprite hinter und welches Sprite vor den Hintergrundzeichen erscheint.

Register 28: Dient zur Umschaltung der einzelnen Sprites in den ein- oder mehrfarbigen Modus.

Register 29: Dieses Register ist dafür zuständig, die einzelnen Sprites in X-Richtung zu vergrößern (wie Register 23, nur X-Achse).

Register 30: Über diese Speicherstelle läßt sich feststellen, welche Sprites sich berühren (Kollisionsregister).

Register 31: Analog hierzu kann man erkennen, ob Sprites mit Hintergrundzeichen kollidieren.

Register 32: Legt die Farbe des Bildschirmrahmens fest.

Register 33: Bestimmt die Farbe des Bildschirmhintergrunds.

Register 37 und 38: Diese Register stellen zwei der insgesamt vier Farben des Sprites im Mehrfarb-Modus (die vierte Farbe ist »transparent«).

Register 39 bis 46: Stellt die dritte Farbe für jedes Sprites im Mehrfarbmodus zur Verfügung.

Bits und Bytes

Bevor wir mit dem Entwurf unseres ersten Sprites beginnen, müssen wir uns zuerst überlegen, wie ein Computer mit Zahlen umgeht. Das mag Ihnen jetzt vielleicht etwas trocken erscheinen, ist aber leider notwendig. Wenn wir Menschen mit Zahlen umgehen, benutzen wir das Dezimalsystem. Im Dezimalsystem existieren 10 Ziffern (0 bis 9), weshalb sich auch der Wert jeder einzelnen Stelle einer Zahl verzehnfacht:

1000	100	10	1	
5	2	8	7	= 5287
3	9	1	4	= 3914

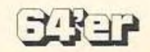
Das ist für uns Menschen leicht zu begreifen, einen Computer stellt es jedoch schon vor größere Probleme, da für ihn nur zwei mögliche Zustände existieren: Entweder fließt Strom oder es fließt kein Strom. Aus diesem Grund benutzen alle Computer das Binärsystem. In diesem Zahlensystem gibt es nur 2 Ziffern (0 und 1). Sie werden sich jetzt fragen: »Wie soll man damit eine halbwegs vernünftige Zahl darstellen ?«. Aber das läßt sich ebenso wie im Dezimalsystem lösen. Der einzige Unterschied besteht darin, daß sich der Wert einer Stelle nicht verzehnfacht, sondern nur verdoppelt:

128	64	32	16	8	4	2	1	
1	0	0	1	1	0	1	1	= 155
0	1	0	0	1	1	0	0	= 76
1	1	1	1	1	1	1	1	= 255

Probieren Sie es aus! Multiplizieren Sie alle Stellen (0 oder 1) einzeln mit dem dazugehörigen Stellenwert (1 bis 128). Wenn Sie nun die Ergebnisse zusammenzählen, so erhalten Sie den dezimalen Wert, der sich aus der Binärzahl errechnet. Wir wissen nun, daß es mit acht binären Stellen möglich ist, Zahlen von 0 bis 255 darzustellen. Diese acht Stellen werden im Computer von einem Byte repräsentiert (ein Byte ist der Inhalt einer Speicherzelle). Eine einzelne dieser Stellen bezeichnet man als Bit (das bedeutet: ein Byte besteht aus acht Bits). Da wir bei der Arbeit mit einem Computer immer

Register	Adresse		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	dezimal	hex								
0	53248	$D000	X-Position Sprite Nr. 0							
1	53249	$D001	Y-Position Sprite Nr. 0							
2	53250	$D002	X-Position Sprite Nr. 1							
3	53251	$D003	Y-Position Sprite Nr. 1							
4	53252	$D004	X-Position Sprite Nr. 2							
5	53253	$D005	Y-Position Sprite Nr. 2							
6	53254	$D006	X-Position Sprite Nr. 3							
7	53255	$D007	Y-Position Sprite Nr. 3							
8	53256	$D008	X-Position Sprite Nr. 4							
9	53257	$D009	Y-Position Sprite Nr. 4							
10	53258	$D00A	X-Position Sprite Nr. 5							
11	53259	$D00B	Y-Position Sprite Nr. 5							
12	53260	$D00C	X-Position Sprite Nr. 6							
13	53261	$D00D	Y-Position Sprite Nr. 6							
14	53262	$D00E	X-Position Sprite Nr. 7							
15	53263	$D00F	Y-Position Sprite Nr. 7							
16	53264	$D010	Sprite 7, msb X-Position	Sprite 6, msb X-Position	Sprite 5, msb X-Position	Sprite 4, msb X-Position	Sprite 3, msb X-Position	Sprite 2, msb X-Position	Sprite 1, msb X-Position	Sprite 0, msb X-Position
17	53265	$D011	msb des Rasterregisters (Reg. 18)	Schaltbit für veränderten Hintergrund-farbmodus 1 = eingeschaltet	Schaltbit für Hochauflösungs-modus 1 = eingeschaltet	Schaltbit für Schirm löschen 0 = gelöscht	Schaltbit für Zeilenzahl 0 = 24 Zeilen 1 = 25 Zeilen	Wert der Zeilenverschiebung in Y-Richtung beim Smooth Scrolling		
18	53266	$D012	Rasterregister. Dazu kommt das msb in Bit 7, Register 17							
19	53267	$D013	Lichtgriffel X-Position							
20	53268	$D014	Lichtgriffel Y-Position							
21	53269	$D015	Ein- und Ausschalten von Sprites. 0 = Sprite aus, 1 = Sprite an							
			Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
22	53270	$D016	(unbenutzt)		Reset-Bit, muß 0 sein, damit VIC-II-Chip arbeitet	Schaltbild für Mehrfarbmodus 1 = eingeschaltet	Schaltbit für Spaltenzahl 0 = 38 Spalten 1 = 40 Spalten	Wert der Spaltenverschiebung in X-Richtung beim Smooth Scrolling		
			Sprite-Vergrößerung in Y-Richtung. 0 = normale Größe, 1 = doppelte Größe							
23	53271	$D017	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
24	53272	$D018	Startadresse Textbildschirm				Startadresse Zeichengenerator oder HiRes-Bitmap			(unbenutzt)
25	53273	$D019	Interrupt-Flag-Register				Lichtgriffel-Interrupt-Flag	Sprite/Sprite-Kollision	Sprite/Hinter-grund-Kollision	Raster-Interrupt-Flag
26	53274	$D01A	Interrupt-Masken-Register				Lichtgriffel-Interrupt	Sprite/Sprite-Kollision	Sprite/Hinter-grund-Kollision	Raster-Interrupt-Maske
			Sprite/Hintergrund-Prioritätenregister. 0 = Sprite hat Priorität, 1 = Hintergrund hat Priorität							
27	53275	$D01B	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
			Sprite-Mehrfarbmodus-Register. 0 = Normaldarstellung, 1 = Mehrfarbmodus-Darstellung							
28	53276	$D01C	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
			Sprite-Vergrößerung in X-Richtung. 0 = normale Größe, 1 = doppelte Größe							
29	53277	$D01D	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
			Sprite/Sprite-Kollision. 0 = keine Berührung. 1 = Berührung							
30	53278	$D01E	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
			Sprite/Hintergrund-Kollision. 0 = keine Berührung, 1 = Berührung							
31	53279	$D01F	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
32	53280	$D020	(unbenutzt)				Farbe des Bildschirmrahmens			
33	53281	$D021	(unbenutzt)				Hintergrundfarbe Nr. 0 (normale Hintergrundfarbe)			
34	53282	$D022	(unbenutzt)				Hintergrundfarbe Nr. 1			
35	53283	$D023	(unbenutzt)				Hintergrundfarbe Nr. 2			
36	53284	$D024	(unbenutzt)				Hintergrundfarbe Nr. 3			
37	53285	$D025	(unbenutzt)				Sprite-Mehrfarben-Register Nr. 0			
38	53286	$D026	(unbenutzt)				Sprite-Mehrfarben-Register Nr. 1			
39	53287	$D027	(unbenutzt)				Sprite 0, Farbe			
40	53288	$D028	(unbenutzt)				Sprite 1, Farbe			
41	53289	$D029	(unbenutzt)				Sprite 2, Farbe			
42	53290	$D02A	(unbenutzt)				Sprite 3, Farbe			
43	53291	$D02B	(unbenutzt)				Sprite 4, Farbe			
44	53292	$D02C	(unbenutzt)				Sprite 5, Farbe			
45	53293	$D02D	(unbenutzt)				Sprite 6, Farbe			
46	53294	$D02E	(unbenutzt)				Sprite 7, Farbe			

Tabelle 1. Alle Register des Video-Chips auf einen Blick



wieder vor die Aufgabe gestellt werden, einzelne dieser Bits ein- oder auszuschalten, soll hier noch etwas über das sogenannte Maskieren gesagt werden. Diese Technik hat nichts mit Karneval zu tun, es bedeutet, daß eine Zahl mit einer anderen Zahl (der sogenannten »Maske«) auf irgendeine Weise verknüpft wird. Eine Verknüpfung ist vergleichbar mit einer Rechenoperation. Bei der Verknüpfung, wie bei einer Rechenoperation, gibt es als Ausgangspunkt zwei Zahlen (die Operanden) und als Ergebnis wiederum eine Zahl. Der Unterschied liegt nun darin, daß bei einer Verknüpfung jedes Bit einzeln betrachtet werden muß (man nennt das bitweise Verknüpfung, also jede Binärstelle für sich). Hier nun die wichtigsten Verknüpfungen, die beim C 64 auch als Basic-Befehle vorhanden sind. Bit 1 und Bit 2 sollen in den nachfolgenden Aufstellungen für die zu verknüpfenden Bits stehen, Bit E repräsentiert das jeweilige Ergebnis.

AND : Bit E = 1 wenn Bit 1 = 1 UND Bit 2 = 1
OR : Bit E = 1 wenn Bit 1 = 1 ODER Bit 2 = 1

1. Beispiel:

Bit-Wertigkeit		128 64 32 16 8 4 2 1	
	Bit 1	1 0 0 1 1 0 0 1	(153)
AND	Bit 2	1 1 0 0 1 1 0 0	(204)
	Bit E	1 0 0 0 1 0 0 0	(136)

2. Beispiel:

Bit-Wertigkeit		128 64 32 16 8 4 2 1	
	Bit 1	1 0 0 1 1 0 0 1	(153)
OR	Bit 2	1 1 0 0 1 1 0 0	(204)
	Bit E	1 1 0 1 1 1 0 1	(221)

Nehmen wir an, Bit 5 soll eingeschaltet werden und der aktuelle Inhalt unseres Bytes ist 153, dann würde das folgendermaßen aussehen:

Bit Nr.		7 6 **5** 4 3 2 1 0	
Bit-Wertigkeit		128 64 **32** 16 8 4 2 1	
	Bit 1	1 0 **0** 1 1 0 0 1	(153)
OR	Bit 2	0 0 **1** 0 0 0 0 0	(32)
	Bit E	1 0 **1** 1 1 0 0 1	(185)

(Falls Sie glauben, die OR-Verknüpfung sei nichts weiter als eine Addition, bedenken Sie bitte, daß es für uns bei der Verknüpfung uninteressant ist, ob das Bit, welches wir einschalten wollen, schon eingeschaltet ist. Bei der Addition hingegen wäre das sehr wichtig.)

Nehmen wir nun an, Bit 5 soll ausgeschaltet werden, und der aktuelle Inhalt unseres Bytes ist 178

Bit Nr.		7 6 **5** 4 3 2 1 0	
Bit-Wertigkeit		128 64 **32** 16 8 4 2 1	
	Bit 1	1 0 **1** 1 0 0 1 0	(178)
AND	Bit 2	1 1 **0** 1 1 1 1 1	(223)
	Bit E	1 0 **0** 1 0 0 1 0	(146)

Wenn wir uns jetzt noch überlegen, wie wir zu der jeweils notwendigen Maske (Zahl, mit der verknüpft wird) gelangen, sollte es für Sie kein Problem mehr sein, einzelne Bits ein- oder auszuschalten. Beginnen wir mit der Maske zur OR-Verknüpfung. Hier müssen die betreffenden Bits in der Maske gesetzt sein, um sie einzuschalten. Folgende Formel ermöglicht das:

MA= 2↑BI + 2↑BI + ... (MA=Maske, BI=Betreffende Bits, ↑=Basic-Befehl Potenzieren)

Da die Bits, die ausgeschaltet werden sollen, nicht, alle anderen Bits jedoch gesetzt werden müssen, muß die Formel so aussehen:

MA=255-(2↑BI+2↑BI+...)

Nachdem Sie sich so lange mit Bits und Bytes beschäftigen mußten, können wir uns nun an unseren ersten Sprite-Entwurf heranwagen.

Entwerfen eines Sprites

In Bild 1 sehen Sie ein Sprite-Entwurfsblatt, das Ihnen beim Entwerfen Ihrer Sprites von großer Hilfe sein wird. Die einzelnen Kästchen stellen die Punkte unseres Sprites dar, welche später auf dem Bildschirm erscheinen oder auch nicht erscheinen werden. Wie Sie feststellen werden, sind jeweils 8 dieser Kästchen durch eine stärkere Linie zusammengefaßt. Wir wissen bereits, daß ein Byte aus 8 Bits besteht, und für eben diese Bits stehen die Kästchen. Auf diese Weise können wir dem C 64 sagen, wie unser Sprite aussehen soll.

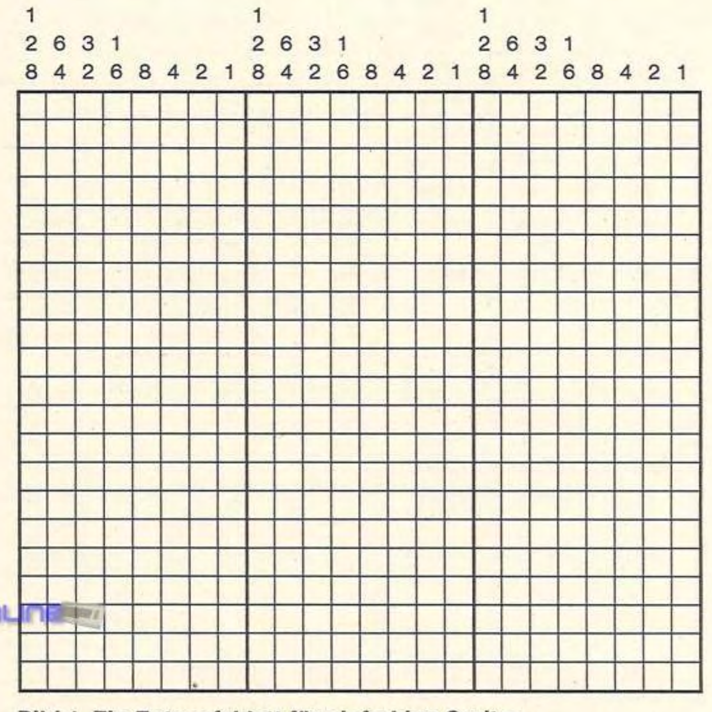


Bild 1. Ein Entwurfsblatt für einfarbige Sprites

Versuchen Sie einmal selbst ein einfaches Sprite zu entwerfen (ein beliebiges Zeichen oder ein Buchstabe). Dazu sollten Sie alle Punkte (bzw. Kästchen), die auf dem Bildschirm erscheinen sollen, ausmalen. Wenn Sie das hinter sich gebracht haben, sehen Sie das sogenannte Bit-Muster Ihres Sprites. Leider können wir dieses Bit-Muster nicht direkt eingeben, aber wir wissen uns zu helfen. Alles, was wir tun müssen, ist die Zusammenfassung von je 8 Bits zu einer Dezimalzahl. Beginnen wir mit den ersten 8 Punkten unseres Sprites. Sie befinden sich auf unserem Sprite-Entwurfsblatt in der linken oberen Reihe. Sie müssen nun jede einzelne Stelle von links nach rechts gehend (die gekennzeichneten Punkte haben den Wert 1, die leeren Punkte den Wert 0) mit dem dazugehörigen Wert, den Sie in der Kopfzeile des Sprite-Entwurfblattes finden, multiplizieren. Nach jeweils 8 Multiplikationen werden die Ergebnisse addiert. Auf diese Weise können Sie jeden 8-Bit-Block in eine Dezimalzahl umrechnen. Wenn Sie so mit allen 63 8-Bit-Blöcken verfahren, erhalten Sie 63 Zahlen, die Sie jetzt in den Computer eingeben können. Die übersichtlichste Lösung ist es, die 63 Zahlen in 21 DATA-Zeilen (zu je 3 Einträgen) unterzubringen. Das könnte etwa so aussehen (natürlich sollten Sie hier Ihre eigenen Werte einsetzen):

```
10 DATA 15, 27,127
11 DATA 0,255, 3
...
30 DATA 14, 27,111
31 DATA 11,169, 55,0
```

Mit den folgenden, zusätzlichen Programmzeilen wird das Sprite dann in den Speicher eingelesen und eingeschaltet:

```
120 FOR AA=832 TO 896
140 READ AB:REM Daten einlesen
160 POKE AA,AB:REM in den Speicher setzen
180 NEXT
200 V=53248 (Basisadresse des VICs)
220 POKE V+21,1:REM Bit 0 wird gesetzt.
                  Sprite 0 soll erscheinen.
230 POKE V+39,1:REM Sprite 0 Farbe weiß
```

Wenn wir unser kleines Programm starten, werden wir feststellen, daß überhaupt nichts passiert. Nun, das Sprite ist da! Wir können es nur noch nicht sehen. Betrachten wir noch einmal Tabelle 1. Die ersten Register bezeichnen die Position unseres Sprites. Da wir in diese Register aber noch nichts hineingeschrieben haben, enthalten sie nur Nullen. Das bedeutet: Unser Sprite erscheint an der Position 0,0 (X/Y). Da diese Stelle aber leider außerhalb des Bildschirms liegt, können wir das Sprite auch nicht sehen. Sagen wir dem VIC also, wohin mit unserem Sprite:

```
240 POKE V+0,150:REM X-Position = 150
241 POKE V+1,120:REM Y-Position = 120
```

Endlich sehen wir etwas. Aber, ist das unser Sprite? Keine Sorge, natürlich ist es das nicht. Die Daten unseres Sprites stehen zwar im Speicher, aber damit kann der Computer noch nichts anfangen, da er noch gar nicht weiß, wo die Daten stehen. Für diese Aufgabe stehen die Speicherstellen 2040 bis 2047 zur Verfügung. Hier müssen wir festlegen, woher die Sprite-Daten stammen sollen (2040 für Sprite 0, 2041 für Sprite 1 ...). Dies geschieht mittels der sogenannten Blocknummer. Ein Block besteht aus 64 Bytes (ein Sprite plus ein Byte ungenutzter Speicher). Wenn wir nun in die Speicherstelle 2040 die Blocknummer 13 schreiben, so erwartet der VIC die Sprite-Daten ab Speicherstelle 13*64=832. Probieren wir das gleich aus:

```
280 POKE 2040,13:REM Daten für Sprite 0
                 stehen in Block 13
```

Voilà, das Sprite sieht endlich so aus, wie wir uns das vorgestellt hatten.

Mehr Platz für Sprites

Wir haben gelernt, daß wir zur Darstellung von Sprites jeweils 64 Bytes freien Speicherplatz benötigen. Freier Speicherplatz bedeutet, daß dieser Speicherplatz nicht anderweitig vom Computer genutzt oder womöglich verändert werden darf. Darin liegt das Problem, über das wir in diesem Abschnitt nachdenken wollen. Die Speicherstellen 0 bis 827 werden als »Rumpelkammer« des Betriebssystems und des Basic-Interpreters benutzt. (Das Betriebssystem und der Basic-Interpreter sind fest in den Computer eingebaute Programme, die dafür sorgen, daß wir etwas in den Computer eingeben können und er das auch verarbeiten kann.) Die Speicherplätze von 828 bis 1029, in dem die Daten für unser Sprite stehen, werden Bandpuffer genannt. Er wird nur benutzt, wenn wir mit einem Kassetten- oder Datenrecorder arbeiten, weshalb wir ihn auch ungestraft belegen können, solange wir keine Datasette benutzen. In den Speicherplätzen 1024 bis 2023 stehen die Zeichen, die auf dem Bildschirm dargestellt werden. Ab Speicherplatz 2048 an aufwärts werden Basic-Programme und Variable gespeichert. Wir sehen, wenn wir mehr als drei verschiedene Sprites benutzen wollen, müssen wir uns um freien Speicherplatz bemühen. Hierzu betrachten wir uns einmal ein paar Pointer aus der Zero-Page (Speicherstellen 0 bis 255). Pointer sind Speicherzellen, in denen verschiedene Adressen abgelegt sind. Diese Pointer (sie werden auch Zeiger genannt) deuten auf bestimmte Bereiche im Speicher unseres Computers. Er braucht die Zeiger, um Daten, welche von ihm gespeichert wurden, auch wiederzufinden. (Wenn wir auf einer Autobahn einen Reifen verlieren würden und uns den Kilometerstein merkten, an dem uns dieses Malheur widerfuhr, so wäre die Nummer des Kilometersteins der Pointer auf unseren Reifen.) Aber nun zurück zu den Pointern in unserem Computer:

Speicherplatz 43 und 44: Zeiger auf die Adresse, an der das Basic-Programm beginnt.

Speicherplatz 45 und 46: Zeiger auf die Adresse, ab der die Variablen gespeichert werden.

Speicherplatz 47 und 48: Zeiger auf die Adresse, ab der die Arrays (das sind DIMensionierte Variablen) angelegt werden.

Speicherplatz 49 und 50: Zeiger auf das Ende der Arrays.

Sie werden sich fragen, weshalb der Computer für eine Adresse je zwei Bytes belegt. Immer wenn der Computer Zahlen benutzt, die größer als 255 sind, verbraucht er zwei Bytes (Low- und High-Byte). Das High-Byte wird mit 256 multipliziert und zum verbleibenden Low-Byte addiert, um eine echte Dezimalzahl zu erhalten. Wenn von Low- und High-Bytes gesprochen wird, so ist immer das erstere das Low- und das letztere folglich das High-Byte. Mittels dieser Technik können wir nun unsere Basic-Programme im Speicher des C 64 an jede beliebige Adresse legen. Die in unserem Fall sinnvollste Adresse ist 16384 (wir können 256 Blöcke zu 64 Byte ansprechen, 256*64=16384). Bevor wir jedoch die Adressen verändern, sollten wir unser Programm auf Diskette oder Kassette abspeichern, da es der Computer sonst nicht mehr finden würde und es somit verloren wäre. Außerdem müssen wir unsere neue Adresse noch in Low- und High-Byte zerlegen. Am einfachsten ist es mit folgender Formel:

HB=INT(DZ/256):LB=DZ-HB*256

(HB=High-Byte, DZ=Dezimalzahl (in diesem Fall die neue Adresse), LB=Low-Byte, INT(...) ist ein Basic-Befehl, der von einer Zahl oder einem Ausdruck innerhalb der Klammern den Teil hinter dem Dezimalkomma abschneidet – nicht rundet! –).

Wir erhalten 0 (Low) und 64 (High). Jetzt soll uns nichts mehr davon abhalten, unser neuerworbenes Wissen auch in die Tat umzusetzen:

POKE 43,0: POKE 44,64:POKE 64*256-1,0:NEW

(NEW setzt alle anderen Pointer automatisch auf den richtigen Wert. Außerdem ist hier wichtig, das letzte Byte vor dem eigentlichen Anfang auf Null zu setzen, da Sie sonst bei jeder Eingabe einen SYNTAX ERROR erhalten würden.)

Jetzt können wir alle Blöcke von 32 bis 63 und von 128 bis 255 verwenden. (Für den Fall, daß Sie trotzdem die Blöcke 64 bis 127 (4096 bis 8192) benutzen, wundern Sie sich bitte nicht über das traurige Bild, das Ihnen Ihre Sprites bieten werden. Wenn diese Blöcke angesprochen werden, greift der VIC nicht auf das Basic-RAM, sondern auf das an für ihn an gleicher Adresse zu liegen scheinende Charakter-ROM zurück, in dem die Form der Buchstaben und Grafikzeichen festgelegt ist.)

Wie stelle ich mehrere Sprites gleichzeitig dar?

Nachdem wir nun im Speicher Platz geschaffen haben, wollen wir einmal mehrere Sprites gleichzeitig auf dem Bildschirm erscheinen lassen.

In Bild 2 haben wir für Sie bereits ein weiteres Sprite entwickelt. Laden Sie bitte unser altes Programm wieder in den Computer und fügen Sie die aus Listing 1 ersichtlichen DATA-Zeilen ein.

Die Daten unserer zwei Sprites sollen nun ab Adresse 2048 abgelegt werden. Hierzu müssen wir nachfolgende Zeilen unseres Programmes folgendermaßen verändern:

```
120 FOR AA=2048 TO 2175
```

(Daten werden nun in Block 32 und 33 gespeichert)



220 POKE V+21,1+2
(Bit 0 und 1 werden gesetzt. Sprites 0 und 1 sollen erscheinen)
242 POKE V+2,120
243 POKE V+3,160 (Sprite 1 soll an die Stelle 120,160 gesetzt werden)
280 POKE 2040,32 (Die Daten für Sprite 0 stehen in Block 32)
281 POKE 2041,33 (Die Daten für Sprite 1 stehen in Block 33)

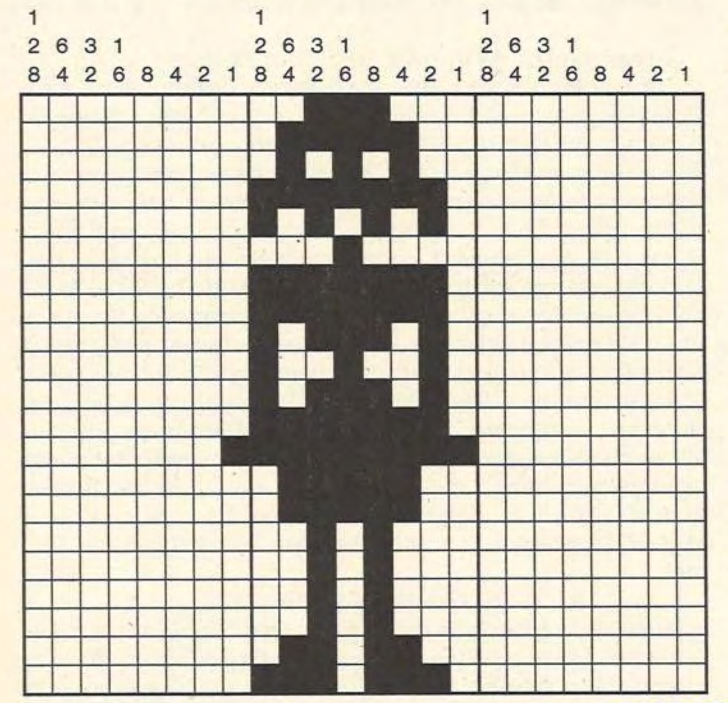


Bild 2. So würde unsere Sprite-Dame aus Listing 1 auf dem Papier aussehen

Wenn Sie unser erweitertes Programm nun starten, sehen Sie, daß beide Sprites auf dem Bildschirm erscheinen. Sollte das nicht der Fall sein, finden Sie in Listing 2 noch einmal das komplette Programm, das Sie noch um Ihre eigenen Sprite-Daten (am besten zwischen Zeilen 8 bis 119) ergänzen müssen.

Jetzt kommt Bewegung in die Sprites

Es ist an der Zeit, uns zu überlegen, auf welche Weise wir die Sprites bewegen können. Betrachten wir noch einmal Tabelle 1. Wenn wir in die Register, mit denen der VIC die Positionen der Sprites auf dem Bildschirm kontrolliert, verschiedene Zahlenwerte schreiben, so bewegt sich das betreffende Sprite. Fügen Sie folgende Zeilen an unser Programm an:

300 BW=BW+1 (Jedesmal wenn unser Programm hier vorbeikommt, wird der Wert der Variable BW um eins erhöht und somit unser Sprite um einen Punkt weiter nach rechts bewegt)

320 POKE V+0,BW (Position wird im X-Positions-Register von Sprite 0 gespeichert)

340 IF BW>320 THEN BW=0 (Sprite soll bis Position 320 bewegt werden)

380 GOTO 300

Nachdem Sie das Programm gestartet haben, wird Sprite 0 von Links nach Rechts über den Bildschirm bewegt. Aber leider wird sich das Programm mit folgender Meldung vorzeitig von uns verabschieden: »?ILLEGAL QUANTITY ERROR IN 320«

Betrachten wir Zeile 320 und den Wert der Variable BW. Der Computer versucht, in Zeile 320 den Wert 256 in eine Speicherstelle zu schreiben. Da ein einzelnes Byte aber nur Zahlen von 0 bis 255 enthalten kann, sind wir geneigt zu glauben, daß unser Computer das Sprite gar nicht bis zum rechten Rand bewegen kann. Diese Annahme ist aber erfreulicherweise falsch. Register 16 macht es möglich, das Sprite über die gesamte Breite unseres Bildschirms zu bewegen. Wir müssen nur dafür sorgen, daß das betreffende Bit in Register 16, welches festlegt ob ein Sprite an Position 0 bis 255 oder an Position 256 bis 320 erscheinen soll, rechtzeitig gesetzt (beziehungsweise zurückgesetzt) wird. Es wäre möglich, dieses Problem mittels einer IF-THEN-Verzweigung in den Griff zu bekommen, eleganter ist es jedoch mit folgender Formel:

BH=INT(BW/256):BL=BW-BH*256 (Das ist die gleiche Formel, welche wir zur Berechnung des Low- und High-Bytes benutzt haben.

BH=Wert des Bits in Register 16, BL=Wert des Registers 0, BW=Position des Sprites)

Fügen Sie noch diese Zeile in Ihr Programm ein:

330 POKE V+16,BH (Bit Nr. 0 wird gesetzt beziehungsweise zurückgesetzt)

Aus Listing 3 (ohne die DATAs und erst ab Zeile 200) können Sie ersehen, wie das Programm nun aussehen muß. Wenn Sie einmal genau hinsehen, werden Sie bemerken, daß das bewegte Sprite nicht sofort auf dem Bildschirm sichtbar wird, vielmehr sieht es so aus, als ob das Sprite von außen unter dem Bildschirmrahmen hindurch in den Bildschirm geschoben wird. Das liegt daran, daß (wir wissen es bereits) der Punkt 0 außerhalb des Bildschirms liegt, da die Position, die wir dem VIC angeben, sich immer auf die rechte untere Ecke bezieht. Um das Sprite nun auch aus dem Bildschirm hinauszubewegen, müssen wir nur den Wert, bis zu dem unser Sprite bewegt werden soll, von 320 auf 344 abändern (ein Sprite ist 24 Punkte, der Bildschirm 320 Punkte breit - 320+24=344). Diesen Wert finden wir in Zeile 340, die demnach folgendermaßen aussehen muß:

340 IF BW>344 THEN BW=0
(Sprite soll bis Position 344 bewegt werden)

Inzwischen dürfte es dem Sprite Nr. 1 (bis jetzt bewegte sich nur Sprite 0) ziemlich langweilig geworden sein, weshalb wir auch sofort daran gehen werden, das zu ändern. Um Sprite 1 für seine Vernachlässigung zu entschädigen, soll es mit doppelter Geschwindigkeit über den Bildschirm bewegt werden. Wenn wir unser jetziges Programm einfach noch einmal eingeben, so müßte dies möglich sein. Ergänzen Sie also das Programm um folgende Zeilen:

305 CW=CW+2 (Position wird jedesmal um 2 erhöht)
315 CH=INT(CW/256):CL=CW-CH*256
325 POKE V+2,CW (Position wird im X-Positions-Register von Sprite 1 gespeichert)
335 POKE V+16,CH*2 (Bit Nr. 1 hat den Wert 2, weshalb CH mit zwei multipliziert wird)
345 IF CW>344 THEN CW=0

Wenn Sie unser Programm nun starten, so benimmt es sich am Anfang recht manierlich, doch warten wir ein wenig. Schon nach kurzer Zeit können sich unsere Sprites nicht mehr entscheiden, wo sie erscheinen sollen. Betrachten wir die Zeilen 330 und 335 unseres Programms. In Zeile 330 schreiben wir in Register 16 den Wert für Sprite 0 (wir beeinflussen Bit 0). Wenn wir nun in Zeile 335 den Wert für Sprite 1 schreiben, so löschen wir den alten Eintrag. Aus diesem Grund müssen wir Register 16 für alle Sprites gleichzeitig ansprechen. Löschen Sie Zeile 335 und ändern Sie Zeile 330 auf folgende Weise ab:

330 POKE V+16,BH*1+CH*2 (Bits 0 und 1 werden gleichzeitig beeinflußt)

Zum Abschluß dieses Kapitels noch ein paar Worte zur Bewegung von Sprites in Y-Richtung. Grundsätzlich wird sie genau wie die Bewegung in X-Richtung gehandhabt (natürlich müssen Sie die Register für die Y-Positionen benutzen).

Das einzige, worauf zu achten ist, ist die Tatsache, daß ein Sprite erst ab Position 50 ganz auf dem Bildschirm zu sehen ist. Außerdem müssen Sie nicht auf ein Überlauf-Register (wie Register 16 für die X-Positionen) achten. Nun aber endlich zu den Prunkstücken unter den Sprites.

Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel [CLR], muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksummer 64 auf Seite 135.

```
0 DATA 0, 56,0,0,124,0,0, 84,0          <148>
1 DATA 0,254,0,0,170,0,0, 16,0          <001>
2 DATA 0,254,0,0,254,0,0,186,0          <134>
3 DATA 0,146,0,0,186,0,0,254,0          <040>
4 DATA 1,255,0,0,124,0,0,124,0          <172>
5 DATA 0, 40,0,0, 40,0,0, 40,0          <132>
6 DATA 0, 40,0,0,108,0,0,238,0,0        <131>
```

Listing 1. Die DATAs zu unserer Sprite-Frau

```
200 V=53248                             <077>
220 POKE V+21,1+2                       <140>
230 POKE V+39,1                         <155>
231 POKE V+40,1                         <122>
240 POKE V+0,150                        <232>
241 POKE V+1,120                        <006>
242 POKE V+2,120                        <039>
243 POKE V+3,160                        <076>
280 POKE 2040,32                        <075>
281 POKE 2041,33                        <142>
300 BW=BW+1                             <191>
310 BH=INT(BW/256):BL=BW-BH*256         <059>
320 POKE V+0,BL                         <122>
330 POKE V+16,BH                        <161>
340 IF BW>320 THEN BW=0                 <153>
380 GOTO 300                            <070>
```

Listing 3. Ihr Sprite beginnt sich zu bewegen

```
140 FOR AA=2048 TO 2303                 <118>
160 READ AB                             <236>
180 POKE AA,AB                          <038>
190 NEXT                                <200>
200 V=53248                             <077>
220 POKE V+28,1                         <049>
230 POKE V+39,6                         <160>
240 POKE V+37,10                        <132>
241 POKE V+38,1                         <102>
250 POKE V+32,11                        <079>
251 POKE V+33,11                        <144>
260 POKE V+0,80                         <029>
261 POKE V+1,80                         <062>
280 POKE 2040,32                        <075>
300 POKE V+21,1                         <193>
```

Listing 5. Das farbige Sprite erscheint

```
320 POKE 2040,33                        <117>
325 GOSUB 500                           <047>
330 POKE 2040,32                        <125>
335 GOSUB 500                           <057>
400 GOTO 320                            <122>
500 BW=BW+4                             <138>
510 BH=INT(BW/256):BL=BW-BH*256         <003>
520 POKE V+0,BL                         <068>
530 POKE V+16,BH                        <107>
540 IF BW>320 THEN BW=0                 <099>
560 FOR W=0 TO 50:NEXT                  <150>
600 RETURN                              <150>
```

Listing 6. Die Bewegung: Das Sprite läuft

```
0 DATA 0, 56,0,0,124,0,0, 84,0          <148>
1 DATA 0,254,0,0,170,0,0, 16,0          <001>
2 DATA 0,254,0,0,254,0,0,186,0          <134>
3 DATA 0,146,0,0,186,0,0,254,0          <040>
4 DATA 1,255,0,0,124,0,0,124,0          <172>
5 DATA 0, 40,0,0, 40,0,0, 40,0          <132>
6 DATA 0, 40,0,0,108,0,0,238,0,0        <131>
7 :                                     <239>
120 FOR AA=2048 TO 2175                 <163>
140 READ AB                             <216>
160 POKE AA,AB                          <018>
180 NEXT                                <190>
200 V=53248                             <077>
220 POKE V+21,1+2                       <140>
230 POKE V+39,1                         <155>
240 POKE V+0,150                        <232>
241 POKE V+1,120                        <006>
242 POKE V+2,120                        <039>
243 POKE V+3,160                        <076>
280 POKE 2040,32                        <075>
281 POKE 2040,33                        <078>
```

Listing 2. Unsere Sprites werden sichtbar. Bitte vergessen Sie nicht, Ihre eigenen Sprite-Daten einzufügen.

```
10 DATA  0,  0,  0,  0,243,192,  3,253, 16 <215>
11 DATA  3,214, 20,  3, 85, 84,  0, 93, 80 <201>
12 DATA  2, 87,  0,  2, 85,  0, 10,165,  0 <111>
13 DATA 10, 10,  0, 10,  2,128, 10,  2,128 <169>
14 DATA  2,133,128,  3,133,  0,  3,247,  0 <197>
15 DATA  3,252,196,  7,243,212,  5,207,212 <098>
16 DATA  5, 81, 84,  1, 81, 80,  1, 81, 64 <144>
19 DATA  0                                 <162>
20 DATA  0,243,192,  3,253,  0,  3,214, 16 <237>
21 DATA  3, 85, 84,  0, 93, 84,  0, 87, 16 <158>
22 DATA  2, 85,  0,  2,165,  0, 10,170,  0 <244>
23 DATA 10,138,  0, 10,130,128, 10,130,128 <043>
24 DATA  2,130,128,  3,130,  0,  3,215,  0 <065>
25 DATA  3,215,  0,  0,252,  0,  0,253,  0 <016>
26 DATA  0, 85, 64,  0, 85, 64,  0, 69, 64 <041>
29 DATA  0                                 <172>
30 DATA  0,  0,  0,  3,207,  0,  4,127,192 <141>
31 DATA 20,151,192, 21, 85,192,  5,117,  0 <153>
32 DATA  0,213,128,  0, 85,128,  0, 90,160 <223>
33 DATA  0,160,160,  2,128,160,  2,128,160 <219>
34 DATA  2, 82,128,  0, 82,192,  0,223,192 <074>
35 DATA 19, 63,192, 23,207,208, 23,243, 80 <218>
36 DATA 21, 69, 80,  5, 69, 64,  1, 69, 64 <052>
39 DATA  0                                 <182>
40 DATA  3,207,  0,  0,127,192,  4,151,192 <225>
41 DATA 21, 85,192, 21,117,  0,  4,213,  0 <180>
42 DATA  0, 85,128,  0, 90,128,  0,170,160 <100>
43 DATA  0,162,160,  2,130,160,  2,130,160 <155>
44 DATA  2,130,128,  0,130,192,  0,215,192 <045>
45 DATA  0,215,192,  0, 63,  0,  0,127,  0 <131>
46 DATA  1, 85,  0,  1, 85,  0,  1, 81,  0 <141>
49 DATA  0                                 <192>
```

Listing 4. Jetzt wird's bunt. DATAs für unser Multicolor-Sprite.

```
310 X=0:Y=0                             <087>
320 P2=PEEK(56320)                      <178>
321 IF(P2 AND 1)=0 THEN Y=-4            <250>
322 IF(P2 AND 2)=0 THEN Y=+4            <060>
323 IF(P2 AND 4)=0 THEN X=-4:XB=2       <025>
324 IF(P2 AND 8)=0 THEN X=+4:XB=0       <188>
330 IF X=0 AND Y=0 THEN 320             <118>
340 POKE 2040,32+XB                     <023>
345 GOSUB 500                           <067>
350 POKE 2040,33+XB                     <035>
355 GOSUB 500                           <077>
400 GOTO 310                            <106>
500 BW=BW+X                             <174>
501 CW=CW+Y                             <212>
505 IF BW>344 THEN BW=0                 <067>
506 IF BW<0 THEN BW=BW+344              <182>
507 IF CW>242 THEN CW=0                 <015>
508 IF CW<0 THEN CW=CW+242              <135>
510 BH=INT(BW/256):BL=BW-BH*256         <003>
520 POKE V+0,BL                         <068>
521 POKE V+1,CW                         <240>
530 POKE V+16,BH                        <107>
560 FOR W=0 TO 50:NEXT                  <150>
600 RETURN                              <150>
```

Listing 7. Sie können das Sprite mit dem Joystick steuern

Die Multicolor-Sprites

Sprites im Mehrfarbmodus (Multicolor-Sprites) können, im Gegensatz zu normalen Sprites, mehrfarbig dargestellt werden. Ein Multicolor-Sprite ist zwar immer noch 24 Punkte breit, wir können aber nur noch 12 Punkte definieren, da sich ein Punkt nun in seiner Breite verdoppelt. Es stehen nun für jeden Punkt, der auf dem Bildschirm erscheinen soll, zwei Bits zur Verfügung und eben diese zwei Bits können wir benutzen, um die jeweilige Farbe, in der der betreffende Punkt erscheinen soll, auszuwählen. Hier nun die möglichen Bit-Kombinationen und die dazugehörige Wirkung. (Denken Sie daran, daß bei Multicolor-Sprites jeweils zwei Punkte zusammengehören. Der eine Punkt bedeutet: Dieser Pixel ist gesetzt. Der zweite Punkt gibt die jeweilige Farbe des Pixels an.):

Bit-Muster	Wirkung
00	Punkt bleibt unsichtbar
01	Punkt erscheint in der Farbe, welche in Register 37 abgelegt wurde
10	Punkt erscheint in der Sprite-Farbe, das heißt, die Farbe wird benutzt, die in Register 39 bis 46 (Sprite 0 bis 7) festgelegt ist
11	Punkt erscheint in der Farbe, welche in Register 38 abgelegt wurde.

Es ist also möglich, ein Sprite aus vier Farben zusammenzusetzen. Wir müssen jedoch bei allen acht Sprites mit zwei gleichen Grundfarben auskommen. In Bild 3 sehen Sie ein Sprite-Entwurfsblatt für Multicolor-Sprites.

Leider ist der Entwurf von Multicolor-Sprites per Hand nicht so einfach wie für normale Sprites, da das Sprite meist anders aussieht als wir uns das ursprünglich vorgestellt hatten. Aus diesem Grund benutzt man am besten einen sogenannten Sprite-Editor, mit welchem das Sprite sofort auf dem Bildschirm dargestellt wird und bei dem das lästige Umrechnen entfällt. (Einen Sprite-Editor, mit dem dies möglich wird, finden Sie ebenfalls in diesem Heft.) Wie Sie bemerken werden, ist nicht mehr jedes einzelne Bit von seinem Nachbarn getrennt, sondern immer zwei Bits zu einem Paar zusammengefaßt. Sie müssen also beim Entwurf eines Sprites darauf achten, daß beide Bits gekennzeichnet werden, wenn Sie einen Punkt erscheinen lassen wollen. Am unteren Rand des Entwurfsblattes finden Sie die bereits erläuterten möglichen Bit-Kombinationen. Die einfachste Lösung wäre, Sie nehmen sich drei Farbstifte (beziehungsweise vier, falls Sie die Hintergrundfarbe auch gleich einzeichnen möchten), und kennzeichnen die einzelnen Bit-Kombinationen mit den dazugehörigen Farben. Wenn Sie nun ein Sprite entworfen und eingezeichnet haben, geht es wieder an das von den normalen Sprites bekannte Umrechnen, wobei darauf zu achten ist, daß für jede Farbe das entsprechende Bit-Muster herangezogen werden muß (man würde sich auch ein wenig schwer tun, wenn man versuchen würde, eine Farbe mit einer Zahl zu multiplizieren). In Listing 4 haben wir für Sie bereits vier Multicolor-Sprites entworfen und in DATA-Zeilen abgelegt. Nachdem Sie das alte Programm gelöscht haben, können Sie die Listings 4 und 5 eingeben.

Obwohl es Ihnen schon vertraut erscheinen sollte, hier noch einmal eine kurze Erläuterung:

Spalten-nummer	0		1		2		3		4		5		6		7		8		9		10		11		Zahlen-codes		
Werte	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1	128	64	32	16	8	4	2	1			
Zeile 0																											
Zeile 1																											
Zeile 2																											
Zeile 3																											
Zeile 4																											
Zeile 5																											
Zeile 6																											
Zeile 7																											
Zeile 8																											
Zeile 9																											
Zeile 10																											
Zeile 11																											
Zeile 12																											
Zeile 13																											
Zeile 14																											
Zeile 15																											
Zeile 16																											
Zeile 17																											
Zeile 18																											
Zeile 19																											
Zeile 20																											

Hintergrundfarbe (transparent)	0	0	Mehrfarben register 0	0	1	Sprite-Farbe	1	0	Mehrfarben-register	1	1

Bild 3. Ein Multicolor-Sprite-Entwurfsblatt

Zeile 140 bis 190: Hier werden die Daten aus den DATA-Zeilen in Block 32 bis 35 gespeichert.

Zeile 220: Sprite 0 wird in den Multicolor-Modus geschaltet, indem das Bit 0 in Register 28 gesetzt wird.

Zeile 230: Sprite 0 wird die Grundfarbe blau zugewiesen.

Zeile 240 und 241: Hier werden die fehlenden zwei Farben festgelegt (sie gelten für alle Multicolor-Sprites, die eingeschaltet werden).

Zeile 250 und 251: Hintergrund und Rahmen des Bildschirms sollen grau sein.

Zeile 260 und 261: Die Position des Sprites wird festgelegt.

Zeile 280: Der Block, in dem die Daten für Sprite 0 liegen, wird angegeben.

Zeile 300: Sprite 0 wird eingeschaltet.

Nachdem Sie das Programm gestartet haben, wird ein kleines Männchen auf dem Bildschirm zu sehen sein. Dieses Männchen ist eines der Multicolor-Sprites, welche wir in den DATA-Zeilen angelegt haben. Es erscheint Ihnen jetzt sicher ein wenig überflüssig, daß wir vier Sprites definieren, aber nur eines sichtbar machen. In wenigen Augenblicken werden Sie sehen, wozu wir vier Sprites brauchen.

Sprites werden lebendig

Geben Sie folgende Zeilen ein und starten Sie unser Programm noch einmal:

```
320 POKE 2040,33
330 POKE 2040,32
400 GOTO 320
```

Unser Männchen scheint auf der Stelle zu gehen. Wie kommt dieser Effekt zustande? In Zeile 320 soll die Form des Sprites 0 aus Block 33 verwendet werden, in Zeile 321 wird jedoch Block 32 benutzt. Indem wir laufend die Form des gleichen Sprites verändern, hat es den Anschein, unser Sprite bewege sich. Um diese Illusion noch zu verstärken, könnte man das ganze Sprite zusätzlich zur Eigenbewegung des Männchens über den Bildschirm schieben. Wir können hierzu das gleiche Programm (beziehungsweise den Programmteil) verwenden, welches wir bereits zur Bewegung des Single-Color-Sprites benutzt haben. Aus Listing 6 ersehen Sie die Programmzeilen, die Sie in das Programm aus Listing 5 einfügen müssen.

Der Programmteil, welcher unser Sprite bewegt, wurde an das Ende des Programms verschoben, da diese Zeilen als Unterprogramm benutzt werden. Die Tatsache, daß wir diese Bewegungsroutine als Unterprogramm anlegen müssen, erklärt sich aus den Zeilen 325 und 335. Um eine kontinuierliche Bewegung zu erzeugen, muß das Sprite nach jeder Veränderung seiner Form verschoben werden. In den Zeilen 325 und 335 sehen Sie diesen Sprung in das Unterprogramm, welches diese Verschiebung vornimmt. In Zeile 560 wurde eine FOR-NEXT-Schleife eingefügt, die den Bewegungsvorgang etwas verlangsamt, jedoch keine weitere Funktion hat. Wenn Sie das Programm nun starten, wird unser kleines Männchen über Ihren Bildschirm spazieren. Sie sehen, schon mit einem Sprite können wir recht possierliche Effekte aus dem Computer zaubern. Bei dieser Gelegenheit können wir uns noch ein wenig mit den Registern 23 und 29 beschäftigen. Mit Hilfe dieser Register ist es möglich, einzelne (oder alle) Sprites zu vergrößern. Geben Sie einmal folgende Zeile in den Computer ein:

```
POKE V+23,1
```

Das Sprite erscheint nun in Y-Richtung gestreckt. Mit der nächsten Zeile verhelfen wir unserem Sprite wieder zu einem »normalen« Äußeren, indem es auch noch in X-Richtung vergrößert wird:

```
POKE V+29,1
```

Diese beiden Register können ohne weitere Probleme gehandhabt werden, Sie müssen nur darauf achten, daß bei jedem Zugriff auf das Register immer an alle Sprites gedacht wird oder die Manipulation eines Bits durch Maskieren vorgenommen wird. Diese Technik, mit der wir uns schon in Kapitel Bits und Bytes beschäftigten, benötigen wir auch für unser nächstes Vorhaben.

Sprite-Steuerung durch den Joystick

In fast allen Computerspielen können wir den Helden des Spieles durch den Joystick lenken. Um dies auch mit unseren hausgemachten Sprites zu realisieren, müssen wir zuallererst einmal unseren Joystick genauer betrachten. Ein Joystick besteht aus fünf Schaltern (oben, unten, links, rechts, Feuerknopf), welche durch Betätigen des Hebels ein- und ausgeschaltet werden. Vereinfacht kann man sagen, wir haben fünf Bits (die Schalter), die wir beeinflussen können, zur Verfügung. In Bild 4 sehen Sie, für welche Bits die einzelnen Schalter zuständig sind.

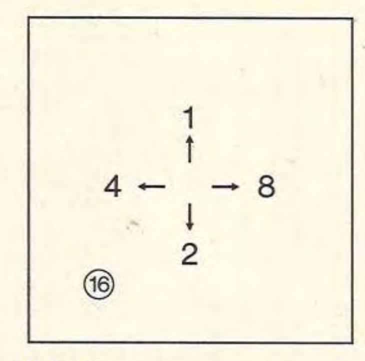


Bild 4. Der Aufbau eines Joysticks. Hier sehen Sie, welche Bits für welche Richtungen durch Maskieren abgefragt werden müssen. Der Wert 16 stellt den Feuerknopf dar.

Diese Bits finden wir im Speicherplatz 56320 für Port 2 und Speicherplatz 56321 für Port 1. Die Bits (Bit 0 bis Bit 4) verhalten sich jedoch ein wenig seltsam. Wenn die Schalter offen sind, haben die betreffenden Bits den Wert »1«, sind sie jedoch geschlossen, ist der Wert »0«, was uns jedoch nicht weiter stören soll. Während wir uns mit der Technik des Maskierens beschäftigten, wurde verschwiegen, daß es auf diese Weise auch möglich ist, abzufragen, ob einzelne Bits gesetzt sind. Wenn wir den zu prüfenden Wert mit einem Operanden, in dem nur das für uns interessante Bit gesetzt ist, mittels einer AND-Operation verknüpfen, so können wir aus dem Ergebnis der Verknüpfung herauslesen, ob das Bit gesetzt ist oder nicht. Oje, das ist wieder einer dieser Sätze, die man dreimal lesen muß, um sie zu verstehen. Aber keine Angst, hier gleich ein Beispiel:

		7	6	5	4	3	2	1	0	
Bit Nr.		7	6	5	**4**	3	2	1	0	
Wert		128	64	32	**16**	8	4	2	1	
	Bit 1	0	1	0	**1**	1	0	1	1	(92)
AND	Bit 2	0	0	0	**1**	0	0	0	0	(16)
	Bit E	0	0	0	**1**	0	0	0	0	(16)

In diesem Fall war das Bit 4 also gesetzt (das Ergebnis würde 0 lauten, wenn Bit zurückgesetzt gewesen wäre). Wenn wir nun in unserem Programm abfragen, in welcher Stellung sich der Joystick befindet, so lassen sich die Sprites in die entsprechenden Richtungen bewegen. Aus Listing 7 können Sie ersehen, wie ein Programm mit Joystick-Abfrage aussehen könnte. Löschen Sie bitte alle Zeilen des aktuellen Programmes, die auf Zeile 300 folgen, und ersetzen sie durch die Zeilen aus Listing 7.

Das Programm besteht nun aus einer Routine zur Joystick-Abfrage, aus den bekannten Zeilen, welche das Sprite verändern sowie aus einem erweiterten Unterprogramm, mit welchem es nun auch möglich ist, das Sprite in Y-Richtung zu bewegen. Hier die detaillierte Erläuterung des Programmes:

Ergänzen Sie jetzt Ihre 64'er-Sammlung

Schaffen Sie sich ein interessantes Nachschlagewerk und gleichzeitig ein wertvolles Archiv!

Kennen Sie alle Ausgaben von 64'er? Suchen Sie einen ganz bestimmten Testbericht? Oder haben Sie einen Teil eines interessanten Kurses versäumt? Suchen Sie nach einer speziellen Anwendung?

Damit Sie jetzt fehlende Hefte mit »Ihrem« Artikel nachbestellen können, finden Sie auf diesen Seiten eine Zusammenstellung aller wesentlichen Artikel der Ausgaben 01 bis 12/85.

Und so kommen Sie schnell an die noch lieferbaren Ausgaben: Prüfen Sie, welche Ausgabe in Ihrer Sammlung noch fehlt, oder welches Thema Sie interessiert. Tragen Sie die Nummer dieser Ausgabe und das Erscheinungsjahr (z.B. 2/85) auf dem Bestellabschnitt der hier eingehefteten Bestell-Zahlkarte ein. Die ausgefüllte Zahlkarte einfach heraustrennen und Rechnungsbetrag beim nächsten Postamt einzahlen. Ihre Bestellung wird nach Zahlungseingang umgehend zur Auslieferung gebracht.

Aktuell

Stichwort	Titel	Seite	Ausgabe
Inhalt	Jahresinhaltsverzeichnis 4/84 bis 3/85	84	04/85
Allgemeines	Commodore Gestern Heute Morgen	10	01/85
Computer	Amiga — Der neue Supercomputer	8	09/85
DFÜ	MCI Mail — Die schnelle Post	8	02/85
Interview	Interview mit David Crane (Game Designer)	146	06/85
Lernen	Schule braucht Computer (VAM-Computer)	9	06/85
Messen	International Chaos Communication Congress	15	03/85
	Heiße Messe in der Wüste: CES	8	03/85
	Die Sportler kommen (CES Software-Bericht)	9	04/85
	Musikmesse Frankfurt	8	04/85
	Hannover-Messe '85	8	06/85
	Hannover-Messe '85	8	07/85
	Chicago im Zeichen der CES	8	08/85
	Aktuelles von der C'85 in Köln	15	08/85
	Btx Total (Internationale Funkausstellung)	8	10/85
	PCW-Computermesse in London	8	11/85
	Neues von der Commodore-Fachausstellung 1985	8	12/85
Recht	Die neue Abmahnmasche — Vorsicht bei Programmangeboten	8	05/85
	Die Ex-Knacker — wo sind sie geblieben?	27	08/85
	Interview mit Raubkopierern (Section 8)	28	08/85
	Schützer kontra Knacki's	23	08/85
	Raub-Talkshow	12	08/85
	Das Urheberrechtsgesetz und Gedanken zu seiner Anwendung	21	08/85
	Änderung des Urheberrechtsgesetzes	162	09/85

Buchbesprechungen

Stichwort	Titel	Seite	Ausgabe
Anfänger	Goldmann Computer Compact	87	03/85
	Basic-Wegweiser für den C 64	86	05/85
	Alles über den C 64, Sachbuchreihe, Band 1	115	06/85
	Lehrspielzeug Computer: C 64/VC 20	112	11/85
	C 64 Computerhandbuch	171	11/85
	Einführungskurs: Commodore 64	144	12/85
Anwendung	Dienstprogramme VC 20, C 64 und SX	86	05/85
	Spaß an Mathe mit dem Commodore 64	88	07/85
	Mathe für die Oberstufe mit dem C 64	88	07/85
	Mathematische Routinen VC 20, Elektrotechnik/Elektronik	112	11/85
	Commodore 64-Listings, Band 2: Dateiverwaltung, Schule, Hobby	112	11/85
	Das Trainingsbuch zum Datamat	144	12/85
C 128	Bücher zum C 128	22	10/85
DFÜ	Das Mailbox-Jahrbuch: Nutz die Netze	112	11/85
Grafik	Grafik auf dem Commodore 64 (+Fehlert. 9/85)	86	05/85
	Einführung in CAD mit dem Commodore 64	128	06/85
	Grafik & Musik auf dem Commodore 64	88	07/85
	Verschiedene Grafikbücher zum C 64	115	08/85
Programmieren	Von Basic zu Assembler: Das Commodore-Buch, Band 4	115	06/85
	64 Intern	115	06/85
	Das Interface Age System-Handbuch zum C 64	115	06/85
	Das C 64 Buch, Band 5: Simons Basic Leitfaden	144	12/85
	Basicode	144	12/85
	Noch mehr Tips und Tricks zum 64er	144	12/85
Speichern	Das Kassettenbuch zum C 64 und VC 20	87	03/85
	Die Floppy 1541 (M&T)	88	07/85
Spiele	Rombachs C 64 Spielführer	87	03/85
	Commodore 64-Listings, Band 1, Spiele	112	11/85
	35 ausgesuchte Spiele für Ihren Commodore 64	171	1/85

64'er Extra

Stichwort	Titel	Seite	Ausgabe
Prozessor	Befehlssatz des 6502/6510-Prozessors	84	09/85
Grafik	Die Videochip-Register des C 64	92	10/85
Sound	Der SID-Chip, seine Register und Programmierung	92	11/85
Speicher	Die Speicherbelegung des C 64	96	12/85

Abenteuerlösungen

Stichwort	Titel	Seite	Ausgabe
Lösungen	Dallas-Quest Lösung	90	01/85
	Lösung The Hobbit	49	02/85
	Guncho Krill-Enchanter ist gelöst	44	03/85
	Infocom-Geheimnisse gelüftet?	49	05/85
	Des Rätsels Lösung: Amazon	145	06/85
	Activision-Adventures entschleiert (Mindshadow, Tracer Sanction)	36	12/85
	Eureka! — ich hab's!	37	12/85
	Lösungen zu Hitchhiker's Guide und Sorcerer	39	12/85

Spiele-Tests

Stichwort	Titel	Seite	Ausgabe
007	James Bond — A View to a Kill	156	09/85
Abenteuer	Abenteuerpaket 1	48	08/85
	Amazon — das besondere Adventure	49	04/85
	Gordon Saga	48	02/85
	Shadowfire	146	09/85
	The Quest — mit C 64 auf Suche nach Drachen	47	01/85
Action	Hexenküche	50	07/85
	Impossible Mission	46	02/85
	Master of the Lamps	48	07/85
	Rescue on Fractalus	158	10/85
	Stellar 7	49	08/85
Construction Set	Mail Order Monsters	49	08/85
	Racing Destruction Set	50	08/85
Geschicklichkeit	Australopedicus Robustus	50	08/85
	Boulder Dash II	159	10/85
	Crystal Castles	50	07/85
	Gribbly's Day out	148	09/85
	Rock'n Bolt	48	08/85
	Thing on a Spring	159	10/85
	Tom + Zaga	48	01/85
Pseudo-Adventures	Roland's Rat Race	49	08/85
	Fourth Protocol und Frankie g.t.H.	162	11/85
Renner	Die Renner 1985: Meistverkaufte Spiele	34	12/85
Schach	Viermal Schachmatt: Verschiedene Schachprogramme	32	12/85
Simulation	Elite	148	09/85
	Jump Jet	148	09/85
	Super Huey Hubschraubersimulator	49	07/85
Sport	Boxspiele: Frank Bruno's B. + Barry McGuigan Champions. B.	49	12/85
	Handkantenschlag per Joystick: Karateka + Exploding Fist	165	11/85
	Nick Faldo Plays the Open (Golf)	159	10/85
	Rallye Speedway	49	07/85
	Slapshot (Eishockey)	50	07/85
	Summer Games II	146	09/85
	World Series Baseball	49	07/85
Diverses	New York City und Air Support	145	06/85

Hardware-Tips und Bauanleitungen

Stichwort	Titel	Seite	Ausgabe
Audio/Video	Besseres Monitorbild beim C 64	90	02/85
	Richtig verbunden — Video/Audio Kabel C 64 (+ Fehlerteufel 3/85)	22	02/85
	Mit 5 Mark zu neuen Dimensionen (Stereoanlage am C 64)	34	05/85
	Ein Monitor ist genug (RGB+Composite an C 128)	16	10/85
C 16	Alte Datasette am C 16	31	04/85
	Alter Joystick am C 16	35	05/85
Eingabegeräte	Der Hexer — Zusatztastatur für den MSE	48	10/85
EPROM	EPROMs im Expansion-Port	46	10/85
	EPROM-Trans — Die Super-Erweiterung	42	10/85
	Das 64'er EPROM-Programmiergerät, Teil 1	44	12/85
Floppy/Datasette	Diskettenlaufwerk 1541 selbst justiert	32	10/85
	Die Datasette streikt nie wieder (Anpassung des Tonkopfs)	34	10/85
IEC-Bus	Auf zu neuen Welten: IEC-Bus im Selbstbau (+Fehlerteufel 10/85)	44	07/85
Joystick	Joystick im Selbstbau	33	03/85
	Dauerfeuer-Adapter	46	08/85
RS232/V.24	Das 30-Mark-Interface (Selbstbau RS232)	29	03/85
	Genau betrachtet: Die RS232/V.24-Schnittstelle	80	05/85
VC 20	16 KByte-Erweiterung umschaltbar	20	02/85
	Der VC 20 steuert Super 8-Kamera	70	02/85
Diverses	Userport-Display	36	05/85
	Reset-Taster für alle Fälle (+Fehlert. 9/85)	130	06/85
	Aus eins mach vier (absturzfreie Betriebssystemumschaltung)	41	07/85

Hardware-Grundlagen

Stichwort	Titel	Seite	Ausgabe
C 16	C 16 — großer oder kleiner Bruder des C 64?	29	04/85
Computer	Was bringt der C 128?	28	11/85
Drucker	Welcher Drucker ist der Richtige? (Grundlagen)	15	05/85
	Hammerwerke — wie funktionieren Typenraddrucker	32	06/85
	Die Alternativen: Thermo-, Tintenstrahldrucker + Plotter	24	07/85
Eingabegeräte	Versteht Sie Ihr Computer? (Wie funktionieren Eingabegeräte)	44	09/85
Floppy	Floppy oder Datasette?	129	06/85
Monitore	Wie funktionieren sie, was ist beim Kauf zu beachten?	16	12/85
	Das Kabel zum Monitor: Welche Normen gibt es?	28	12/85
Peripherie	Grafikeingabegerät: Wie funktionieren sie?	30	08/85

Hardware-Tests

Stichwort	Titel	Seite	Ausgabe
80 Zeichen	Mit 80 fängt das Leben an (Test 80-Zeichen-Karten) (+Fehlert. 6/85)	17	04/85
Computer	Generationswechsel: Test C 16	16	01/85
	Plus und Minus beim Plus/4	14	02/85
	PC 128 — der Profi (Hardwaretest)	13	04/85
	Erster ausführlicher Test C 128 PC (Teil 1)	16	06/85
	Erster ausführlicher Test C 128 PC (Teil 2)	17	07/85
DFÜ	Marktübersicht Modems & Akustikkoppler	32	07/85
Drucker	Vergleich: Drucker unter 700 Mark (Tests und Marktübersicht)	18	05/85
	Tests und Marktübersicht Typenraddrucker	35	06/85
	Test: Brother EP 44	27	07/85
	Brother TC-600	118	08/85
	Riteman C+	133	09/85
	Panasonic KX-P1091	134	09/85
	Star SG 10C	132	09/85
	Melchers CP-80X — wie hätten Sie's denn gern?	25	10/85
	Geheimtip: Der RFI DP 165	24	10/85
	Epson GX 80 — einer für alle	26	10/85
	MPS 803 — ein Drucker für alle Gelegenheiten?	40	1/85
	Epson JX-80 das vielfarbige Druck-Genie	38	11/85
	Epson FX-85 neue Referenz	42	11/85
	SP 1000 VC — Superstar mit Haken	41	11/85
	Der NEC-P2 — das fernöstliche Wunder	159	12/85
	DMPG9 — eine solide Sache	162	12/85
Eingabegeräte	Der Bildschirm wird zur Leinwand (TechSketch-Lightpen)	21	04/85
	Das Doppelleben des Joystick-Ports: 10er-Tastaturen	50	09/85
	Joysticks: Test und Marktübersicht (+Fehlerteufel 12/85)	19	11/85
	Es geht auch anders: Lightpens und Trackballs	22	11/85
EPROMer	Frisch gebrannt ist halb gespeichert (EPROM-Programmiergeräte im Test)	39	07/85
	QuickByte II — das Kraftpaket	14	10/85
Floppy/Datasette	Schnell wie der Wind (Test: Speeddos, Turbo-Access)	22	04/85
	Turbo-Floppies, zweite Generation: Speeddos plus + Prologic DOS	28	10/85
	Das große Rennen: Schnelle Bandlaufwerke	37	10/85
	Professionelle Floppylaufwerke für den C 64 (IEC-Floppies)	30	10/85
	Gut gekauft ist halb gespeichert (Marktübersicht Disketten)	38	10/85
Grafik	Die Videowerkstatt (Digitizer-Test)	32	05/85
	Digitalbilder m.d. C 64: PrintTechnik Digitizer	24	01/85
Interface	Hardware-Interface ganz weich: Test EC 64	23	01/85
	Gute Connections — Übersicht Schnittstellen	21	03/85
	Card/Print +6 — Das Allround-Interface	20	03/85
	Das Wiesemann-Centronics-Interface	18	03/85
	Erst ein IEC-Bus öffnet Tür und Tor (+Fehlert.4/6-85)	24	03/85
Monitore	Eine klare Sache: Test Phönix-Monitor	28	04/85
	Marktübersicht: Monochrome Monitore	30	12/85
Musik	Die Stimme des Meisters: Test Voice Master	19	02/85
	Trommelwirbel: Test Digital Drums	45	08/85
	Die Musikhardware zum C 64	17	09/85
Oszilloskop	Der C 64 als Speicheroszilloskop	26	04/85
Roboter	Roboter selbst gebaut (Fischertechnik)	167	10/85
Scanner	So lernt Ihr Drucker lesen	30	06/85
Speicher	Speichertuning VC 20: Test 64 KByte Karte	26	01/85
Steuern	Flottes Türmchen: MEA-Interface	116	08/85

Kurse

Stichwort	Titel	Seite	Ausgabe
Assembler	Assembler ist keine Alchimie, Teil 5	142	01/85
	Assembler ist keine Alchimie, Teil 6	134	02/85
	Assembler ist keine Alchimie, Teil 7	124	03/85
	Assembler ist keine Alchimie, Teil 8	138	04/85
	Assembler ist keine Alchimie, Teil 9	138	05/85
	Assembler ist keine Alchimie, Teil 10	127	07/85
	Assembler ist keine Alchimie, Teil 11	126	08/85
	Assembler ist keine Alchimie, Teil 12	109	09/85
	Assembler ist keine Alchimie, Teil 13 (Schluß)	143	10/85
C 128	Entdeckungsreise duch den C 128	42	12/85
Comal	Comal — Eine Einführung, Teil 3	130	02/85
Effektives Programmieren	Müllabfuhr im Computer: Garbage Collection, Teil 1	122	01/85
	Stringprogrammierung in Maschinensprache, Teil 2	147	02/85
	Finden mit System, eine neuartige Suchmethode, Teil 3	148	03/85
	Sortieren mit dem Computer, Teil 1	148	04/85
	Sortieren mit dem Computer, Teil 2	159	05/85
	Sortieren mit dem Computer, Teil 3	124	06/85
	Sortieren mit dem Computer, Teil 4	138	08/85
	Sortieren mit dem Computer, Teil 5	124	09/85
	Sortieren mit dem Computer, Teil 6 (Schluß)	150	12/85
Extern	C 64 extern — Der Weg nach draußen, Teil 1	144	08/85
	C 64 extern — Der Weg nach draußen, Teil 2	122	09/85
	C 64 extern — Der Weg nach draußen, Teil 3 (Schluß)	129	10/85
Floppy	In die Geheimnisse der Floppy eingetaucht, Teil 4	148	01/85
	In die Geheimnise der Floppy eingetaucht, Teil 5	130	03/85
	In die Geheimnise der Floppy eingetaucht, Teil 6	145	05/85
	In die Geheimnisse der Floppy eingetaucht, Teil 7 (Schluß)	116	06/85
	Directory-Manipulationen I	140	06/85
Floppy	Directory-Manipulationen II	163	10/85
Grafik	Hires 3 — die Grafikerweiterung zum Grafikkurs, Teil 1	123	02/85
	Hires 3 — 15 neue Basic-Befehle, Teil 2	136	03/85
	Hires 3 — Grafikkurs-Anwendung, Teil 3 (Schluß)	152	08/85
	Sprites ohne Geheimnisse	40	08/85
	Streifzüge durch die Grafikwelt, Teil 1	106	09/85
	Streifzüge durch die Grafikwelt, Teil 2	149	11/85
Logeleien	Logeleien, Teil 1	143	07/85
	Logeleien, Teil 2	136	08/85
	Logeleien, Teil 3 (Schluß)	115	09/85
Musik	Dem Klang auf der Spur, Teil 2	136	01/85
	Dem Klang auf der Spur, Teil 3	152	02/85
	Dem Klang auf der Spur, Teil 4	131	04/85
	Dem Klang auf der Spur, Teil 5	152	06/85
	Dem Klang auf der Spur, Teil 7	132	07/85
	Dem Klang auf der Spur, Teil 8	133	08/85
	Dem Klang auf der Spur, Teil 9	126	10/85
	Dem Klang auf der Spur, Teil 10 (Schluß)	157	11/85
Speicher	Memory Map mit Wandervorschlägen, Teil 3	126	01/85
	Memory Map mit Wandervorschlägen, Teil 4	150	02/85
	Memory Map mit Wandervorschlägen, Teil 5	144	03/85
	Memory Map mit Wandervorschlägen, Teil 6	144	04/85
	Memory Map mit Wandervorschlägen, Teil 7	120	06/85
	Memory Map mit Wandervorschlägen, Teil 8	140	07/85
	Memory Map mit Wandervorschlägen, Teil 9	129	08/85
	Memory Map mit Wandervorschlägen, Teil 10	112	09/85
	Memory Map mit Wandervorschlägen, Teil 11	133	10/85
	Memory Map mit Wandervorschlägen, Teil 12	145	11/85

Software-Tips

Software-Grundlagen

Listings zum Abtippen

Software-Tests

So machen's andere

Zeile 310: Die Variablen, welche die Richtung angeben, in die das Sprite bewegt werden soll, werden gelöscht. Hiermit wurde erreicht, daß das Sprite nur bewegt wird, wenn der Joystick benutzt wird.

Zeile 320: In der Variable P2 wird der Wert der Speicherstelle 56320 (Zustand von Joystick-Port 2) abgelegt.

Zeile 321 bis 324: Alle vier Bits werden nacheinander getestet. Ist das Bit nicht gesetzt (der dazugehörige Schalter im Joystick geschlossen), wird in die Variable X oder Y der Wert geschrieben, um den das Sprite verschoben werden soll (positive Werte für Bewegung Rechts/Abwärts, negative Werte für Bewegung Links/Aufwärts). Variable XB kümmert sich darum, ob Sprite-Daten aus Block 32/33 (Männchen rechts) oder Block 34/35 (Männchen links) stammen.

Zeile 330: Wird der Joystick nicht bewegt, so kehrt das Programm zu Zeile 320 zurück.

Zeile 340 bis 400: Hier wird die Blocknummer der Sprites verändert (wie im vorhergegangenen Programm) und das Unterprogramm angesprungen.

Zeile 500 und 501: Die Bewegungswerte werden zu den Positionsvariablen addiert.

Zeile 505 bis 508: Wandert das Sprite aus dem Bildschirm hinaus, wird die betreffende Variable so manipuliert, daß das Sprite auf der gegenüberliegenden Bildschirmseite wieder auftaucht.

Zeile 510: Die bereits bekannte Formel zur Berechnung des Bit-Wertes in Register 16.

Zeile 520 bis 530: Die Position des Sprites wird in die betreffenden Register geschrieben.

Zeile 560: Verzögerungsschleife.

Wird das Programm nun gestartet, so können wir das Männchen auf dem ganzen Bildschirm umhergehen lassen (Falls Sie unser Männchen nicht sehen können, so liegt das daran, daß die Positionsvariablen zu Beginn des Programms den Wert 0 enthalten und somit das Sprite nicht sichtbar ist. Drücken Sie Ihren Joystick nach unten rechts und sofort kommt unser Männchen in den Bildschirm gelaufen).

Zum Abschluß noch ein paar Worte zu Register 27. Stoppen Sie bitte das Programm und löschen Sie den Bildschirm. Gehen Sie jetzt bitte mit dem Cursor etwa in die Mitte des Bildschirms und schreiben Sie ein paar Zeichen (am besten mit CTRL und 9 und dann Leerzeichen) auf den Bildschirm. Wenn Sie nun mit unserem Männchen an den Zeichen vorbeilaufen, so wird es davor erscheinen. Sobald aber in Register 27 das Bit für Sprite 0 gesetzt wird, muß das Sprite hinter den Zeichen vorbeilaufen. Probieren Sie es aus:
`POKE V+27,1` (Sprites untereinander erscheinen übrigens auch nicht alle auf der gleichen Ebene. Sprite 0 wird als erstes dargestellt, die restlichen Sprites erscheinen dahinter. Also Sprite 1 hinter Sprite 0 und so weiter.) Mit Register 27 ist es möglich, einen gewissen 3D-Effekt zu erreichen. Sie könnten das Sprite zum Beispiel um unsere Bildschirmzeichen herumlaufen lassen, wobei Sie nur rechtzeitig Register 27 ansprechen müssen. Für den Fall jedoch, daß es vielleicht an einer Mauer haltmachen muß, hält der VIC eine weitere Überraschung für uns bereit.

Sprite-Kollision

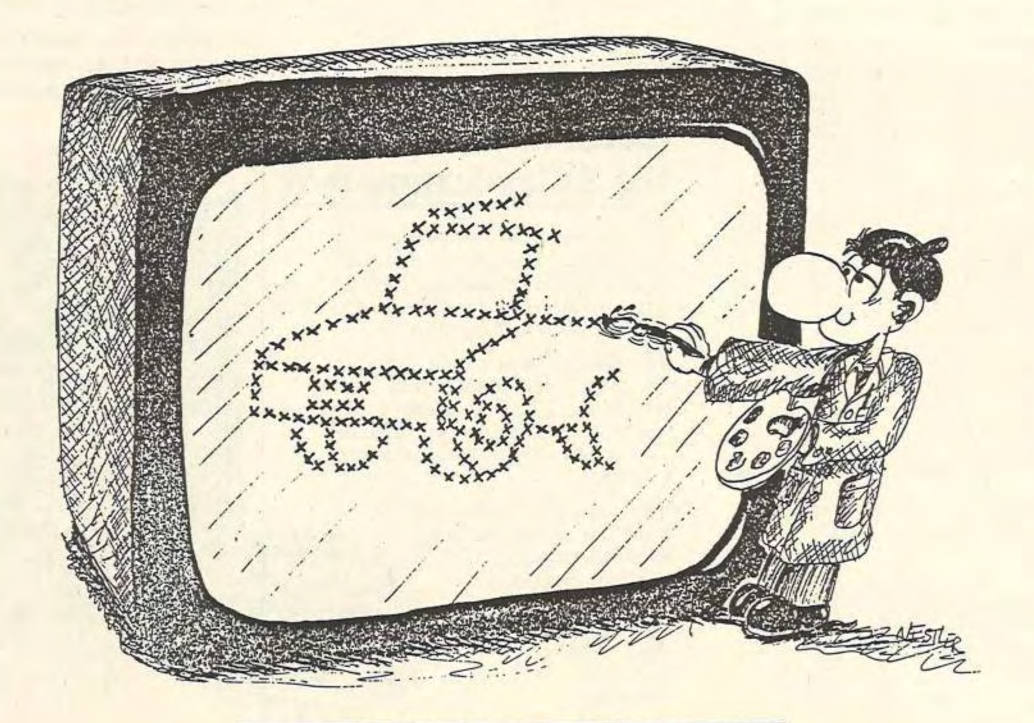
Wie bereits erwähnt, ist es möglich festzustellen, ob sich Sprites untereinander, beziehungsweise Sprites und Hintergrundzeichen berühren. Da Computerspiele ohne diese Abfrage so gut wie undenkbar sind, soll hier erläutert werden, wie die Register, die die Kollision von Sprites und Hintergrundzeichen anzeigen, gehandhabt werden. Sobald sich zwei Sprites berühren (wobei es wichtig ist, daß der VIC nur die sichtbaren Punkte der Sprites beachtet), werden in Register 30 automatisch die betreffenden Bits gesetzt (berühren sich Sprite 2 und Sprite 5, so wird Bit 2 und 5 gesetzt, der Zahlenwert des Registers wäre folglich 36). Register 31 wird analog zu Register 30 behandelt, mit der Ausnahme, daß die Sprite/Hintergrund-Kollision ausschlaggebend ist. Diese beiden Register werden sofort zurückgesetzt, sobald das Ereignis, welches die Bits einschaltete, nicht mehr zutrifft (die Sprites haben sich zum Beispiel bereits aneinander vorbeibewegt). Um diesen Effekt auch in unserem Programm einzusetzen, geben Sie bitte folgende Zeile ein:

```
580 IF PEEK(V+31)<>0 THEN BW=BW-X:CW=CW-Y:GOTO 505
```

(Wenn das Sprite mit einem Zeichen kollidiert, werden die Positionsvariablen wieder auf den ursprünglichen Wert zurückgesetzt).

Mit dieser Programmerweiterung können Sie nun bereits ein eigenes Spiel (zum Beispiel einen Irrgartenlauf) programmieren. Sie haben jetzt die wichtigsten Techniken kennengelernt, mit denen die Sprite-Handhabung möglich wird. Versuchen Sie ruhig, noch etwas mit den Registern rumzuspielen. Kaputtmachen können Sie ja im Video-Chip nichts. Wer weiß, vielleicht schreiben Sie bald ein eigenes Spielprogramm mit Sprite-Steuerungen.

(Thomas Erhart/dm)



Dateiverwaltung für Einsteiger

Eine der sinnvollsten Einsatzmöglichkeiten eines Computers besteht darin, ihn Daten verwalten zu lassen. In einem ausführlichen Kurs möchten wir Ihnen erklären, wie man problemlos Dateiverwaltungen schreibt, und auf welche Dinge Sie bei der Programmierung achten müssen.

Wenn Sie auf Ihrem C 64 nicht nur Fertigprodukte einsetzen, sondern auch selbst programmieren wollen, müssen Sie sich zwangsläufig auch mit der Verwaltung von Daten im Hauptspeicher und vor allem auf einem externen Massenspeicher (Floppy oder Datasette) auseinandersetzen.

In diesem Artikel werden wir Ihnen zeigen, welche Probleme hierbei immer wieder auftreten und wie sie gelöst werden können. Fortgeschrittene C 64-Anwender weise ich darauf hin, daß bei der Speicherung von Daten nur sequentielle Dateien erläutert werden, da diese für Einsteiger bereits problematisch genug sind. Sogenannte »relative« oder »Direktzugriffsdateien« werden nicht berücksichtigt.

Im Handbuch des C 64 wird beschrieben, wie Daten beliebiger Art auf Diskette und Kassette mit Hilfe sogenannter »sequentieller« Dateien verwaltet werden können. Die ersten Versuche mit sequentiellen Dateien führen jedoch meist sehr schnell zur Verzweiflung, da die im Handbuch abgedruckten Beispielprogramme zwar einwandfrei laufen, nach der Abänderung der Programme (Sie wollen schließlich Ihre eigenen Daten verwalten!) meist jedoch nicht mehr funktionieren.

Es gibt C 64-Besitzer, die sich aus lauter Verzweiflung völlig von der Dateiverwaltung abgewendet haben. Ein Leser programmierte sogar eine Adressenverwaltung, indem er ein Programm schrieb, in dem die Adressen in Form von REM-Zeilen enthalten waren:

```
10 rem gerd maier, ottoweg 5, 6700
   ludwigshafen, 0621/36276
20 rem willi schneider, lange-roetter-str.3,
   6800 mannheim, 0621/239672
```

Die Programmzeilen waren alphabetisch nach den Namen geordnet, so daß er bei der Suche nach einer bestimmten Telefonnummer in etwa wußte, welche Programmzeilen er zu LISTen hatte. Es ist zweifellos möglich, auf diese Weise, bei der die Daten unmittelbar im Programm enthalten sind, Daten zu verwalten. Die Daten können aktualisiert werden, indem Programmzeilen eingefügt, gelöscht oder geändert werden.

Der Einsatz eines Computers ist bei dieser Form der Datenverwaltung jedoch vollkommen überflüssig. Das gleiche Ergebnis kann mit Hilfe eines Notizblocks oder eines Karteikastens weitaus einfacher und komfortabler erreicht werden. Die eigentlichen Vorteile der »computerisierten« Dateiverwaltung, das komfortable Eintragen, Löschen, Ändern und vor allem Suchen von Daten kann nicht verwirklicht werden, wenn das Programm und die zu verwaltenden Daten eine Einheit bilden.

Üblicherweise trennt man daher die eigentlichen Daten vom Programm zur Datenverwaltung. Im Programm selbst sind die Daten nicht enthalten, es dient nur zur Steuerung der Arbeit mit der Datei. Die Daten werden getrennt in einer Datei gespeichert, entweder auf Diskette oder auf Kassette. Die Arbeit mit einer solchen Datei verläuft dann wie folgt:

1. Laden und Starten des Dateiverwaltungsprogramms
2. Einlesen der Datei durch das Steuerungsprogramm
3. Arbeit mit der Datei (Suchen, Ändern etc.)
4. Speichern der Datei, da die Daten während der Arbeit eventuell geändert wurden

Organisation der Daten im Computerspeicher

Wenn die eigentlichen Daten nicht unmittelbar in den Programmzeilen enthalten sind, müssen sie in Form von Variablen im Speicher des C 64 abgelegt werden. Das heißt, beim Eintragen einer Adresse werden diese Daten vom Steuerungsprogramm einer Variablen zugewiesen und beim Speichern der Datei wird der Inhalt dieser Variablen gespeichert, um bei der nächsten Programmbenutzung wieder eingelesen zu werden.

Theoretisch ist es nun vollkommen Ihnen überlassen, welche Variablen Sie verwenden, um Daten zu speichern. Eine Möglichkeit wäre zum Beispiel die Verwendung von unterschiedlichen Variablennamen für verschiedene Adressen:

```
10 rem eingabe der adressen
20 input "erste adresse (name)";a1$
30 input "erste adresse (telefonnr.)";a2$
40 input "zweite adresse (name)";b1$
50 input "zweite adresse (telefonnr.)";b2$
60 input "dritte adresse (name)";c1$
70 input "dritte adresse (telefonnr.)";c2$
```

Wenn Sie dieses Programm eingeben und starten, werden Sie nacheinander nach der ersten, zweiten und der dritten Adresse gefragt, und zwar jeweils nach Name und Telefonnummer. Nach der Eingabe befinden sich die Adressen in den Variablen »a1$«, »a2$« bis »c1$«, »c2$«, sie wurden diesen Variablen zugewiesen. Diese Art der Datenspeicherung ist leider außerordentlich unflexibel. Angenommen, Sie suchen die Telefonnummer von Herrn Maier und zur Suche wird der folgende Programmteil verwendet:

```
110 rem suche nach telefonnr.
120 input "name";n$
130 if n$=a1$ then print a2$
140 if n$=b1$ then print b2$
150 if n$=c1$ then print c2$
```

Diese »Suchroutine« wird die gestellte Aufgabe erfüllen und die gewünschte Telefonnummer ausgeben. Auf diese Weise zum Beispiel 100 Adressen zu verwalten, bedeutet jedoch, ein Programm zu schreiben, in dem sowohl in der Eingabe- als auch in der Suchroutine hundertmal der prinzipiell gleiche Befehl vorkommt. Der einzige Unterschied betrifft den jeweiligen Variablennamen. Ein solches Programm zu schreiben, ist sicherlich außerordentlich ineffizient, abgesehen davon, daß Ihnen nach kurzer Zeit die Variablennamen ausgehen dürften.

Die in praktisch jeder Dateiverwaltung verwendete Lösung besteht in der Verwendung indizierter Variablen oder auch sogenannter »Arrays« oder »Felder«. Im folgenden werden wir immer Stringarrays verwenden, da in Stringvariablen Daten beliebiger Art gespeichert werden können, im Gegensatz zu numerischen Variablen.

Auf die Variablen kommt es an

Wie Sie im C64-Handbuch nachlesen können, besitzen Arrayvariablen gleiche Variablennamen und werden durch den Index unterschieden. »a$(1)« ist demnach eine andere Variable als »a$(2)« und von dieser völlig unabhängig. Komfortabel wird die Verwendung von Arrays erst durch die Möglichkeit, als Index selbst eine Variable angeben zu können, zum Beispiel »a$(i)«. Welche Variable tatsächlich angesprochen wird, hängt vom Wert der Variablen »i« ab. Hat »i« den Wert drei, entspricht der Ausdruck »a$(i)« dem Ausdruck »a$(3)«.

Beachten Sie bitte die bei der Verwendung von indizierten Variablen notwendige Dimensionierung mit dem DIM-Befehl, das heißt die Angabe der Arraygröße. Im folgenden werde ich immer von einer Datei ausgehen, in der maximal 500 Adressen gespeichert werden. Entsprechend wird die Dimensionierung vorgenommen. Die Eingabe- und die Suchroutine können nun auf die Verwendung eines Stringarrays umgestellt werden:

```
1  rem dimensionierung
2  dim n$(500),t$(500)
10 rem eingaberoutine v.1
20 for i=1 to 3
30 input "adresse (name)";n$(i)
40 input "adresse (telefonnr.)";t$(i)
50 next
```

Die Programmschleife (Zeile 30 bis 50) wird dreimal abgearbeitet. Beim ersten Schleifendurchgang besitzt »i« den Wert eins. Da die Ausdrücke »n$(i)« und »t$(i)« daher den Ausdrücken »n$(1)« und »t$(1)« entsprechen, werden der vom Benutzer eingegebene Name und die Telefonnummer in den Array-Variablen mit dem Index eins gespeichert.

Beim nächsten Durchgang besitzt »i« den Wert zwei. Die nächste einzugebende Adresse wird daher in »n$(2)« und »t$(2)« gespeichert, und entsprechend die dritte einzugebende Adresse beim letzten Schleifendurchgang in »n$(3)« und »t$(3)«.

Durch Änderungen des Schleifenendwertes – zum Beispiel »for i=1 to 100« – können mit dem gleichen Programm beliebig viele Adressen vom Benutzer eingegeben werden. Diese Routine ist daher weitaus kürzer und flexibler als es bei der Verwendung verschiedener Variablennamen der Fall war. Die entsprechend geänderte Routine zur Suche einer bestimmten Telefonnummer:

```
110 rem suche nach telefonnr. v.1
120 input "name";n$
130 for i=1 to 3
140 if n$=n$(i) then print t$(i)
150 next
```

Wie Sie sehen, ist der Ablauf dieser Routine mit der Eingaberoutine »verwandt«. Auch hier wird eine Schleife benutzt, um mit Hilfe des Schleifenindizes »i«, der bei jedem Durchgang um eins erhöht wird, das gesamte Stringarray »n$(1)« bis »n$(Schleifenende)« zu bearbeiten. Wird eine Übereinstimmung zwischen dem »Suchkriterium«, das heißt dem Namen »n$« und dem momentan bearbeiteten Namen »n$(i)« festgestellt, gibt das Programm die zur jeweiligen Adresse zugehörige Telefonnummer »t$(i)« auf dem Bildschirm aus.

Beachten Sie die »Parallelität« der beiden Stringarrays: Ein bestimmter Index entspricht einer Adresse, die in zwei verschiedenen Stringarrays gespeichert ist. Zu jeder Variablen des Arrays »n$(...)« gibt es eine Variable aus dem Array »t$(...)« mit dem gleichen Index, die die zugehörige Telefonnummer enthält (siehe Bild 1).

Komfortable Dateiverwaltungen in Basic sind nur mit Hilfe indizierter Variablen zu verwirklichen. Da auch in den folgenden Programmbeispielen ständig Array-Variablen verwendet werden, ist es unbedingt notwendig, daß Sie das Prinzip der Indizierung verstehen. Bei Verständnisschwierigkeiten bitte die entsprechenden Kapitel im C64-Handbuch nachlesen.

In der Praxis werden Sie »Name« und »Telefonnummer« sicher nicht als vollständige Adresse empfinden. Das dargestellte Prinzip läßt sich jedoch problemlos auf Daten beliebiger Art übertragen. Nehmen wir als Beispiel eine Schallplattendatei, die die Informationen »Titel«, »Interpret«, »Platte« und »Musikrichtung« enthalten soll (siehe Bild 2). Wir benötigen zur Speicherung der Informationen vier Stringarrays:

t$(...) = Titel
i$(...) = Interpret
p$(...) = Platte
m$(...) = Musikrichtung

Die entsprechend geänderten Programmteile:

```
10  rem eingabe v.2
20  for i=1 to 5
30  input "titel";t$(i)
40  input "interpret";i$(i)
50  input "platte";p$(i)
60  input "musikrichtung";m$(i)
70  next
110 rem suche v.2
120 input "titel";t$
130 for i=1 to 5
140 if t$=t$(i) then print i$(i):print p$(i):
    print m$(i)
150 next
```

Obwohl es sich in diesem Beispiel um eine völlig andere und detailliertere Datei handelt, ist das Prinzip der Dateneingabe und der Suche exakt das gleiche wie zuvor.

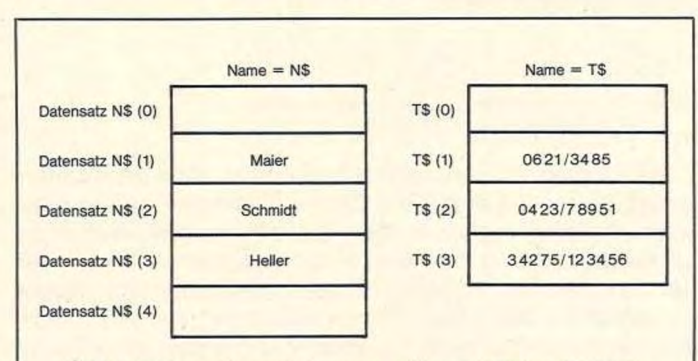


Bild 1. Datensatzspeicherung mit zwei String-Feldern

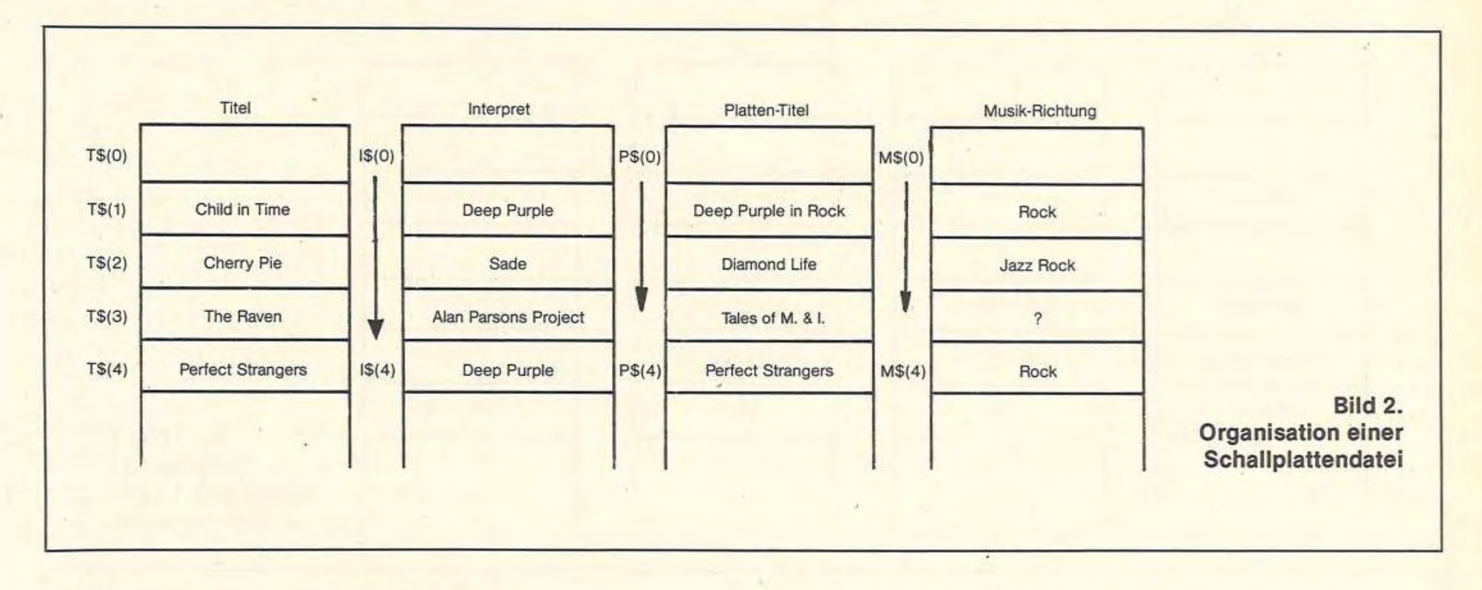


Bild 2. Organisation einer Schallplattendatei

Eintragen von Datensätzen

Die vorgestellten Programmbeispiele sind zwar zur Darstellung des prinzipiellen Umgangs mit Daten sehr gut geeignet, für den Einbau in ein Dateiverwaltungsprogramm bedarf es jedoch noch größerer Änderungen. Eine Hauptforderung an eine Dateiverwaltung ist Flexibilität. In einem vernünftigen Programm dieser Art ist es zum Beispiel undenkbar, daß zwar ein Titel eingegeben und nach dem zugehörigen Interpreten gesucht werden kann, jedoch nicht umgekehrt nach den von einem bestimmten Interpreten geschriebenen Stücken. Die Suchmöglichkeiten sollten so umfangreich wie möglich sein. Jeder (!) Teil des gesamten Datensatzes sollte als Suchkriterium verwendet werden können, so daß zum Beispiel auch ein Plattentitel eingegeben werden kann und alle Stücke auf dieser Platte mit Titel, Interpret und Musikrichtung ausgegeben werden.

Bevor wir versuchen, die Suchroutine entsprechend diesen Ansprüchen umzuschreiben, will ich ein anderes Problem behandeln. Sowohl in der Eingabe- als auch in der Suchroutine werden feste Schleifenstart- und Endwerte benutzt (eins und fünf in Zeile 20 und 120). Wenn wir zum Beispiel 20 Schallplatten mit insgesamt 322 Liedern besitzen, werden wir die Dimensionierung (»dim t$(322),...«) und die Schleifenendwerte (»for i=1 to 322«) entsprechend ändern.

Bei der Eingabe der Datei wird anschließend von uns verlangt werden, alle 322 Datensätze einzugeben, eine kaum an einem Tag zu lösende Aufgabe. Und selbst wenn wir uns diese Mühe machen, ist es eine äußerst ungünstige Lösung, wenn der Zukauf einer weiteren Platte eine Änderung des Programms (FOR-NEXT-Schleife) erforderlich macht.

Die Eingaberoutine muß daher verallgemeinert werden. Nach dem Starten des Programms sollte es möglich sein, beliebig viele Datensätze in die Datei einzutragen. Um keine Daten zu überschreiben, müssen neue Daten an das Ende der jeweiligen String-Felder »angehängt« werden.

Wir benötigen daher eine Variable, die uns ständig Auskunft darüber gibt, wieviele Datensätze sich bereits in der Datei befinden und die wir »ad« (Anzahl der Datensätze) nennen werden. Diese Variable muß immer dann um den Wert eins erhöht werden, wenn ein neuer Datensatz eingetragen wird, und sie muß selbstverständlich zusammen mit der eigentlichen Datei auf Diskette beziehungsweise Kassette gespeichert werden, wenn wir die Arbeit mit dem Programm beenden.

Wenn das Programm geladen und gestartet wird, wird außer der eigentlichen Schallplattendatei auch diese Variable »ad« geladen, so daß unser Dateiverwaltungsprogramm immer »weiß«, wieviele Datensätze sich in der Datei befinden.

```
1 rem dimensionierung
2 dim t$(500),i$(500),p$(500),m$(500)
10 rem eingabe v.3
20 ad=ad+1:rem anzahl datensaetze
30 input "titel";t$(ad)
40 input "interpret";i$(ad)
50 input "platte";p$(ad)
60 input "musikrichtung";m$(ad)
70 input "weiterer Datensatz (j/n)";a$
80 if a$ = "j" then goto 20
```

Die Schleife zur Eingabe der Datensätze entfällt. Bei jeder Benutzung der Eingaberoutine wird ein einzelner Datensatz eingetragen, und zwar »hinter« den bereits vorhandenen Datensätzen.

Bei der ersten Benutzung des Programms besitzt »ad« den Wert null und erhält wegen Zeile 20 den Wert eins. Der erste Datensatz wird daher den Variablen »t$(1)«, »i$(1)«, »p$(1)« und »m$(1)« zugewiesen. Eine weitere Eingabe bewirkt die Erhöhung von »ad« um eins, der zweite Datensatz wird daher in »t$(2)« bis »m$(2)« eingetragen. Jeder einzutragende Datensatz wird automatisch in der nächsten noch nicht belegten Array-Variablen eingetragen.

Wie Sie sehen, gestattet erst die Verwendung indizierter Variablen und die Verwendung einer weiteren Variablen als Index die Programmierung flexibler Dateiverwaltungen, in denen die Anzahl der Datensätze und weitere sogenannte »Dateiparameter« nicht von vornherein starr festgelegt sind.

Löschen von Datensätzen

Nach dem Eintragen von Datensätzen (hier: Titel mit allen weiteren Informationen) wenden wir uns nun dem Gegenteil zu, dem Löschen eines kompletten Satzes. Das Löschen eines Datensatzes kann mit dem Vernichten einer Karteikarte verglichen werden, nur daß anstelle der Karteikarte Variablen »vernichtet« beziehungsweise gelöscht werden.

Die einfachste Möglichkeit, zum Beispiel den dritten Datensatz zu löschen, besteht darin, alle zugehörigen Array-Variablen als leer zu definieren:

```
t$(3) = "":rem titel löschen
i$(3) = "":rem interpret löschen
p$(3) = "":rem plattentitel löschen
m$(3) = "":rem musikrichtung löschen
```

Wenn wir auf diese Weise vorgehen, besitzt unsere Datei nach vielen Löschvorgängen ein Aussehen gemäß Bild 3, es ähnelt einem Schweizer Käse und weist Unmengen von »Löchern« beziehungsweise Lücken auf. Da neue Datensätze prinzipiell am Ende der Datei eingetragen werden, bleiben diese Lücken unbenutzt. Daher wird irgendwann der Fall eintreten, daß unsere Datei voll ist, das heißt die letzten reservierten indizierten Strings mit dem Index 500 belegt sind,

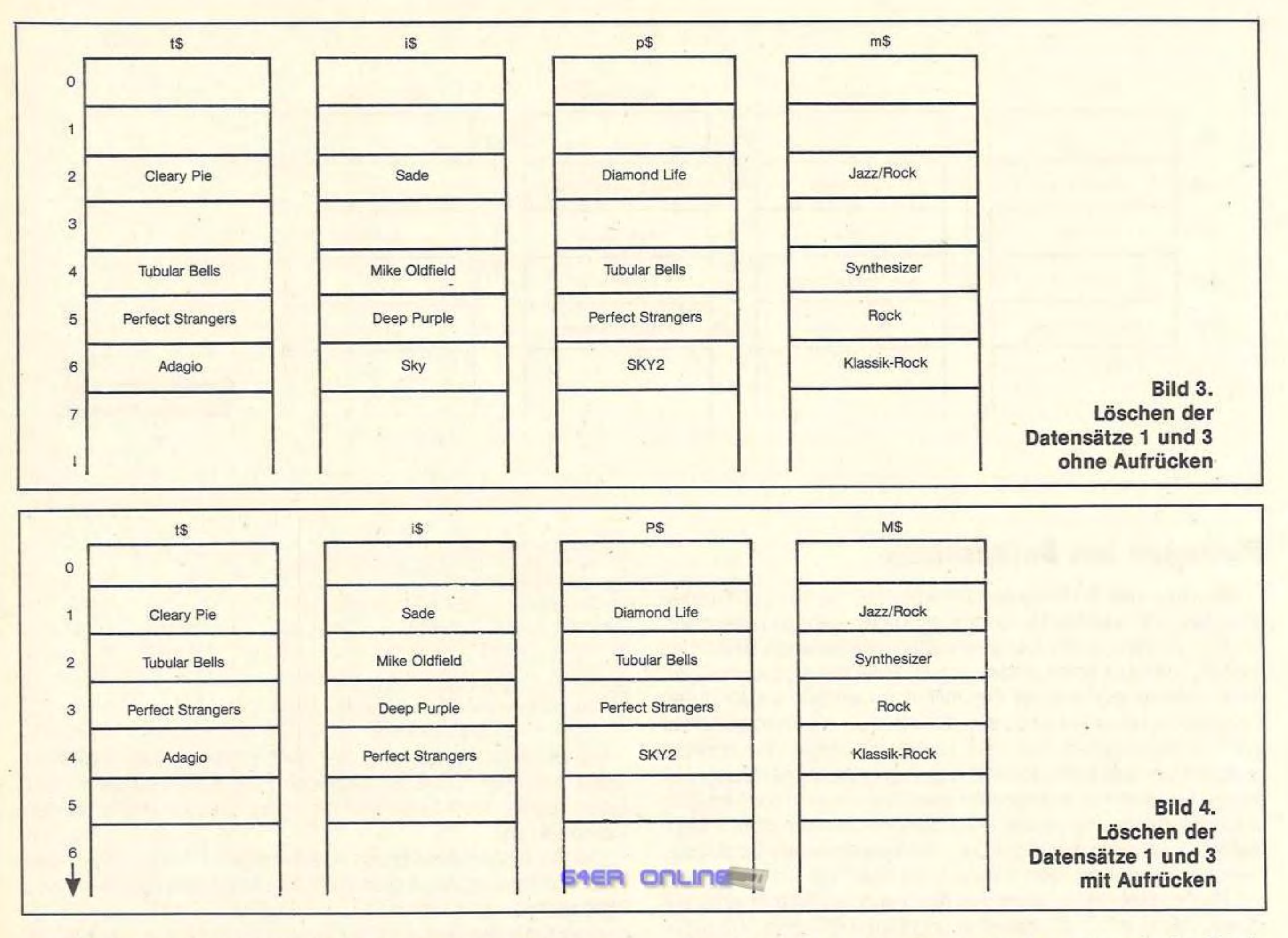


Bild 3. Löschen der Datensätze 1 und 3 ohne Aufrücken

Bild 4. Löschen der Datensätze 1 und 3 mit Aufrücken

obwohl die Datei tatsächlich wesentlich weniger Datensätze enthält.

Beim Löschen von Datensätzen sollte aus diesem Grund die Entstehung von Lücken vermieden werden. Diese Bedingung kann am einfachsten erfüllt werden, wenn alle folgenden Datensätze aufgerückt werden (siehe Bild 4).

```
10 rem loeschen des n-ten satzes
20 for i=n to ad-1
30 t$(i)=t$(i+1)
40 i$(i)=i$(i+1)
50 p$(i)=p$(i+1)
60 m$(i)=m$(i+1)
70 next
80 ad=ad-1
```

Diese Löschroutine ist allgemein genug aufgebaut, um in einer Dateiverwaltung Verwendung zu finden. Gelöscht wird kein starr festgelegter Datensatz, sondern der durch die Variable »n« bestimmte Satz.

Das Programm kann jederzeit durch den Abschluß mit einer RETURN-Anweisung als Unterprogramm formuliert werden. Wenn zum Beispiel der dritte Datensatz gelöscht werden soll, ruft das Hauptprogramm das Unterprogramm auf, nachdem es zuvor der Variablen »n« den Wert drei zugewiesen hat (»n=3:gosub (Datensatz löschen)«).

Ändern und Suchen von Datensätzen

Mit einer Dateiverwaltung, die die Funktionen »Eintragen von Datensätzen« und »Löschen von Datensätzen« besitzt, ist auch das Ändern eines Satzes möglich, indem dieser zuerst komplett gelöscht und anschließend – mit den entsprechenden Änderungen – neu eingetragen wird. Eine zusätzliche Routine ist nicht erforderlich, wenn die Unterprogramme »Löschen« und »Eintragen« geschickt verwendet werden.

Das Suchen von Datensätzen wurde an einem in der Praxis relativ unbrauchbaren Programmbeispiel erläutert. Eine vernünftige Suchroutine sollte es, wie bereits erwähnt, erlauben, einen beliebigen Suchbegriff, das heißt, einen beliebigen Datensatzteil einzugeben.

Eine solche Suchroutine muß daher derart gestaltet werden, daß sie jedes (!) Feld eines Datensatzes mit dem eingegebenen Suchbegriff vergleicht. Entspricht eines dieser Felder dem Suchbegriff, muß der komplette Datensatz angezeigt werden. Negativ an dieser Methode bleibt zu vermerken, daß sie relativ zeitaufwendig ist.

Beachten Sie, daß die Suche nach der Ausgabe des ersten gefundenen Datensatzes keinesfalls abgebrochen werden darf. Ein Beispiel: In einer Adreßdatei wurde als Suchbegriff vom Benutzer der Name »Maier« eingegeben. Die Datei wird nun vom Programm durchsucht und der Datensatz »Maier, Georg, Wilbertweg 5, 6000 München« ausgegeben. Möglicherweise befinden sich in der Datei jedoch mehrere »Maier« und der gefundene Datensatz ist trotz Übereinstimmung mit dem Suchkriterium nicht jener, dessen Adresse vom Benutzer gesucht wird.

Da dieser Fall in der Praxis häufig vorkommt, sollte das Programm den Benutzer nach der Ausgabe eines Datensatzes immer fragen, ob die Datei nach einem weiteren mit dem Suchkriterium übereinstimmenden Satz durchsucht werden soll, oder ob die Suche beendet ist.

Nachfolgend soll eine Suchroutine vorgestellt werden, die allen geschilderten Anforderungen genügt:

```
10 rem suchen v.3
20 input "suchkriterium";s$
30 for i=1 to ad
40 if s$<>t$(i) and s$<>i$(i) and s$<>p$(i)
```




```
    and s$<>m$(i) then 90
50  print t$(i):print i$(i):print p$(i):print m$(i)
60  print "weitersuchen (j/n) ?"
70  get a$:if a$="" then 70
80  if a$="n" then 100
90  next
100 return
```

Kern dieser Suchroutine ist die Schleife (Zeilen 30 bis 90) mit der Schleifenvariablen »i«, die beim ersten Durchgang den Wert eins und beim letzten den Wert »ad« besitzt, entsprechend der Anzahl vorhandener Datensätze.

Mit dieser Schleife wird die Datei Satz für Satz untersucht, indem jedes Teilfeld eines zu untersuchenden Datensatzes mit dem eingegebenen Suchkriterium »s$« verglichen wird.

Wenn keines der Datensatzfelder mit dem Suchbegriff in »s$« identisch ist, wird der nächste Satz untersucht (Zeile 40), ansonsten wird der gefundene Datensatz komplett ausgegeben (Zeile 50) und der Benutzer gefragt, ob die Datei weiter durchsucht werden soll (Zeile 60).

Zeile 70 besteht aus einer sogenannten »Eingabewarteschleife«, die auf einen Tastendruck des Benutzers wartet. Drückt er die Taste »n«, es soll also nicht weitergesucht werden, wird die Suchschleife verlassen und zu Zeile 100 gesprungen, die zur Rückkehr in das Hauptprogramm führt.

Will der Benutzer das Programm hingegen weitersuchen lassen, wird – ebenso wie im Fall des negativen Vergleichs – der nächste Datensatz mit dem Suchkriterium »s$« verglichen. Diese Suchroutine ist bereits weitaus komfortabler als die zuvor besprochene Version, erfüllt jedoch noch nicht die Anforderungen, die an eine komfortable Dateiverwaltung gestellt werden. Wenn Sie sich nach Lektüre dieses Artikels an Ihren C64 begeben, versuchen Sie zumindest, diese Routine um die Möglichkeit zu erweitern, Suchbegriffe abgekürzt eingeben zu können. So daß das Programm nach Eingabe von »Ma*« zum Beispiel alle »Maier«, »Mayer« etc. findet: if right$(s$,1) = "*" then s$ = left $ (s$,len(s$)-1)

Die Suchroutine müßte dann so geändert werden, daß sie die Länge des Suchbegriffs berücksichtigt (len(s$)).

Eintragen, Löschen, Ändern und Suchen

```
10  rem dimensionierung
20  dim t$(500),i$(500),p$(500),m$(500)

100 rem menue
110 print chr$(147):rem bildschirm loeschen
120 print "(e)intragen von datensaetzen"
130 print "(s)uchen von datensaetzen"
140 print "(b)eenden der Arbeit"
150 print:print "kommando?"
160 get a$:if a$="" then 160
170 if a$="e" then gosub 310
180 if a$="s" then gosub 410
190 if a$="b" then end
200 goto 110

300 rem eintragen
310 ad=ad+1:rem anzahl datensaetze
320 input "titel";t$(ad)
330 input "interpret";i$(ad)
340 input "platte";p$(ad)
350 input "musikrichtung";m$(ad)
360 return

400 rem suchen
410 input "suchkriterium";s$
420 for i=1 to ad
430 if s$<>t$(i) and s$<>i$(i) and s$<>p$(i) and
    s$<>m$(i) then 530
440 print t$(i):print i$(i):print p$(i):print m$(i)
450 print "weitersuchen (j/n) ?"
460 get a$:if a$="" then 460
470 if a$="j" then 530
480 print "(l)oeschen/(a)endern ?"
490 get a$:if a$="" then 490
500 if a$="l" or a$="a" then n=i:gosub 610:
    rem loeschen
510 if a$="a" then gosub 310:rem eintragen
520 goto 540:rem ende
530 next
540 return

600 rem loeschen
610 for i=n to ad-1
620 t$(i)=t$(i+1)
630 i$(i)=i$(i+1)
640 p$(i)=p$(i+1)
650 m$(i)=m$(i+1)
660 ad=ad-1
670 return
```

Das Programm zur Verwaltung von Dateien im Speicher des C64 ist nun vollständig und entspricht zum größten Teil den erläuterten Unterprogrammen.

Die größte Änderung betrifft das Auswahlmenü am Programmanfang, das die Bedienung des Programms erleichtern soll und je nach eingegebenem Kommando zu dem entsprechenden Programmteil verzweigt (Zeile 100 bis 220). Ein Kommando wird ausgewählt, indem Sie die Taste betätigen, die dem jeweiligen Anfangsbuchstaben entspricht (»N« für »Neue Daten«, »S« für »Suchen«, »E« für »Ende«).

Sollten Sie die Kommandos »Löschen« und »Ändern« vermissen, so betrachten Sie bitte den Programmteil »Suchen« ab Zeile 400 etwas näher. Beide Funktionen wurden in die Suchroutine integriert. Der Grund: Sowohl zum Löschen als auch zum Ändern eines Datensatzes muß dieser zuerst gesucht werden (welcher Datensatz soll gelöscht/geändert werden?). Sobald ein Satz entsprechend dem Suchkriterium gefunden wurde, und zwar der richtige Datensatz, das heißt, wenn der Benutzer nicht weitersuchen will, wird er gefragt, ob er diesen Satz, der auf dem Bildschirm ausgegeben wurde, löschen oder ändern will (Zeile 480).

Das Programm wartet nun auf einen Tastendruck des Benutzers (Zeile 490). Mit »l« wird der Datensatz gelöscht. Dies geschieht dadurch, daß der Löschroutine – die bekanntlich Satz Nummer »n« löscht – der Index des gefundenen Satzes in der Variablen »n« übergeben und diese anschließend aufgerufen wird (Zeile 500).

Das Ändern des gefundenen Datensatzes verläuft analog, nach dem Löschen wird jedoch zusätzlich das Unterprogramm zum Eintragen eines Datensatzes aufgerufen (Zeile 510) und der Benutzer kann den geänderten Datensatz neu eingeben. Wie bereits gesagt, kann die Funktion »Ändern« mit den beiden Funktionen »Löschen« und »Eintragen« realisiert werden.

Die Leerzeilen im abgedruckten Listing dienen übrigens nur der optischen Strukturierung, um das Programm übersichtlicher zu gestalten und haben keine weitere Funktion.

Sie wissen nunschon eine ganze Menge, was zur Dateibehandlung im Speicher des C64 notwendig ist und können Daten eintragen, löschen, ändern und suchen. Die zugrundeliegenden Prinzipien können Sie sich anhand des Programmbeispiels und der Erläuterungen noch einmal im Zusammenhang verdeutlichen.

Dateiverwaltung ohne das Speichern und Laden der Datei auf externen Speichermedien ist jedoch nicht denkbar. Diesen zugegebenermaßen nicht ganz einfachen Vorgängen werden wir uns nun stellen müssen.

Sequentielle Dateien

Die sogenannten »sequentiellen Dateien« stellen die einfachste Speicherart dar. Mit der Datasette kann ausschließlich diese Dateiart verwendet werden, mit der Floppy können zusätzlich »relative« oder »Direktzugriffsdateien« verwaltet werden, auf die wir jedoch nicht näher eingehen werden. Um diese Dateitypen zu erklären, müßte man sehr weit ausholen, und dies würde den Rahmen dieses Kurses sprengen.

»Sequentiell« bedeutet soviel wie »nacheinander«. Zum Verständnis sequentieller Dateien ist es außerordentlich wichtig, zwischen sequentieller *Datenspeicherung* (Bild 5) und sequentiellem *Datenzugriff* (Bild 6) zu unterscheiden. Für unsere Arrays wurde die sequentielle Datenspeicherung verwendet, die Datensätze wurden fortlaufend gespeichert, hintereinander aufgereiht, so daß keine Lücken in der Datei entstanden. Wir konnten auf diese Daten jedoch jederzeit direkt zugreifen, zum Beispiel mit dem Befehl »print t$(3)«, der gezielt auf das dritte Element mitten in dem Array zugreift und anzeigt.

In sequentiellen Dateien sind die Daten ebenfalls auf dem Speichermedium, der Diskette beziehungsweise Kassette, lückenlos hintereinander aufgereiht, eben sequentiell gespeichert. Im Gegensatz zum Computerspeicher oder den bereits erwähnten Direktzugriffsdateien gestatten sequentielle Dateien jedoch keinen direkten Zugriff auf bestimmte Daten. Wenn wir den dritten Satz einer sequentiellen Datei lesen wollen, müssen wir zuerst den ersten, dann den zweiten Satz lesen, bevor der gewünschte Datensatz eingeladen werden kann. Ebenso verhält es sich mit dem Speichern von Daten: Wenn ein Datensatz mitten in der Datei geändert werden soll, muß die komplette Datei geladen werden, der betreffende Datensatz – nun im Computerspeicher in einem Array abgelegt – wird geändert, und zuletzt wird die komplette geänderte Datei wieder gespeichert. Es ist leider unmöglich, gezielt mitten in einer sequentiellen Datei Lese- oder Schreibvorgänge durchzuführen.

Bild 6. Beim sequentiellen Datenzugriff auf einen beliebigen Datensatz müssen alle vorhergehenden überlesen werden.

Eine Analogie mit Musikstücken auf Kassette bietet sich an: Stellen Sie sich vor, auf einer Kassette haben Sie eine komplette Schallplatte aufgenommen, von der Sie nun das dritte Lied abspielen wollen. Die Kassette ist zum Anfang zurückgespult worden und Sie wollen nun zum betreffenden Lied vorspulen. Leider stellen Sie fest, daß die Taste zum Vorspulen defekt ist. Sie können auf das dritte Lied nun nur noch sequentiell zugreifen, indem Sie zuvor das erste und zweite Lied anhören, das heißt, indem Sie die gesamte »Datei« ab dem »Dateianfang« bis zum gewünschten Lied durchblättern. Analog verläuft die Arbeit mit sequentiellen Dateien.

Somit dürfte der prinzipielle Umgang mit sequentiellen Dateien geklärt sein: Zu Beginn der Arbeit mit der Datei wird diese komplett in den Speicher des C64 eingelesen. Nun kann mit Hilfe der Dateiverwaltung mit den Daten gearbeitet werden, es können Datensätze neu eingetragen, gesucht, gelöscht und geändert werden.

Wenn Änderungen in der Datei vorgenommen wurden, muß (!) die Datei komplett gespeichert werden, wenn die Arbeit beendet werden soll. Keinesfalls darf der Computer einfach ausgeschaltet werden!

Nach diesen mehr theoretischen Ausführungen werden wir uns mit der Praxis des Lesens und Schreibens von Daten mit Hilfe sequentieller Dateien beschäftigen.

Die Datenkanäle

Lese- und Schreibzugriffe auf sequentielle Dateien erfordern eine gewisse Vorbereitung, das »Öffnen der Datei« oder auch »Öffnen eines Datenkanals«. Stellen Sie sich diesen Vorgang am besten analog zu einem Telefongespräch nach Amerika vor, der das Freigeben einer Leitung zum Teilnehmer am anderen Apparat, in diesem Fall zur Datasette oder Floppy, erfordert. Beispiel: OPEN 2,8,3,"Datei,s,r"
oder allgemein:

OPEN LFN,GN,SA,"Name,Typ,Zugriff"

Der zugehörige Befehl ist ziemlich umfangreich. Zuerst muß eine sogenannte »logische Filenummer« (LFN) angegeben werden. Diese LFN ist eine Zahl zwischen 1 und 255. Der Grund: Mehrere Dateien können gleichzeitig geöffnet werden, zum Beispiel um aus einer Datei Adressen zu lesen und gleichzeitig alle Adressen im Raum München in eine zweite Datei zu schreiben (ist allerdings nur mit der Floppy möglich).

Durch Angabe der LFN kann bei den folgenden Lese- beziehungsweise Schreibbefehlen auf die gewünschte Datei Bezug genommen werden; unserem C64 muß schließlich mitgeteilt werden, welche von mehreren geöffneten Dateien gemeint ist.

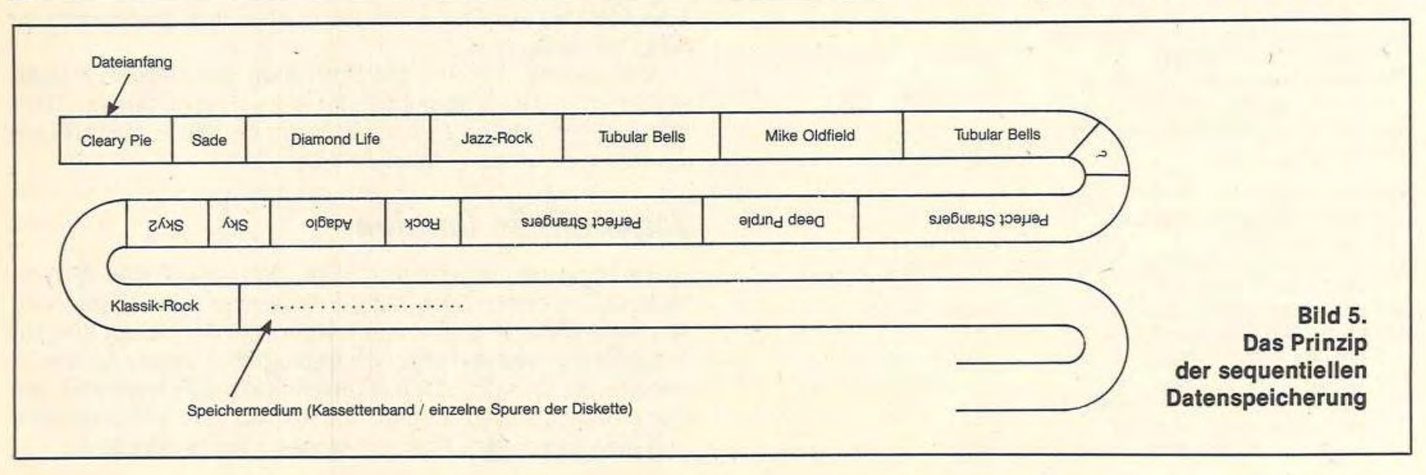


Bild 5. Das Prinzip der sequentiellen Datenspeicherung



Außer der LFN muß beim Öffnen einer Datei die Gerätenummer (GN) mitgeteilt werden, um dem C 64 anzugeben, ob die Datenübertragung zu einem Drucker (GN=4), zur Datasette (GN=1), zur Floppy (GN=8) oder anderen Geräten erfolgen soll.

Der dritte Parameter ist die sogenannte »Sekundäradresse« (SA), eine Zahl zwischen null und 15. Bei Verwendung der Datasette hat die Sekundäradresse eine gewichtige Bedeutung: SA=0 heißt, daß Daten vom Band gelesen und SA=1, daß im folgenden Daten auf das Band geschrieben werden sollen.

Der letzte Parameter ist der »Dateiname« (DN), der bei Verwendung der Datasette nicht angegeben werden muß, in den folgenden Programmbeispielen zur Vereinheitlichung jedoch immer (!) angegeben wird (zudem können aus Besitzern einer Datasette jederzeit Floppy-Benutzer werden).

Wie erläutert, wird bei Verwendung der Datasette mit Hilfe der Sekundäradresse angegeben, ob Lese- oder Schreibzugriffe erfolgen sollen (beides gleichzeitig ist mit der gleichen Datei nicht möglich!). Bei Verwendung einer Floppy dient hierzu der Dateiname, dem entweder die Zeichen »,S,R« folgen müssen, um Lesezugriffe anzugeben (S=sequentiell, R=Read (Lesen)), oder aber die Zeichenfolge »,S,W« für Schreibvorgänge (W=Write (Schreiben)).

Allgemein lautet der Befehl zum Öffnen einer Datei:
OPEN (LFN),(GN),(SA),"(DN)"

Beachten Sie bitte, daß nach Beenden der Arbeit mit einer Datei diese unbedingt mit dem Befehl »CLOSE (LFN)« wieder geschlossen werden muß, da sonst – vor allem beim Schreiben in die Datei – Daten verloren gehen können!

Beispiele:

1. Schreiben in die Datei »Test« (Datasette)

```
10 open 1,1,1,"test":rem datei zum
   schreiben oeffnen
20 rem schreibbefehle
30 close 1:rem datei schliessen
```

2. Lesen aus der Datei »Test« (Datasette)

```
10 open 1,1,0,"test":rem datei zum lesen
   oeffnen
20 rem lesebefehle
30 close 1:rem datei schliessen
```

3. Schreiben in die Datei »Test« (Floppy)

```
10 open 1,8,2,"test,s,w":rem datei zum
   schreiben oeffnen
20 rem schreibbefehle
30 close 1:rem datei schliessen
```

4. Lesen aus der Datei »Test« (Floppy)

```
10 open 1,8,2,"test,s,r":rem datei zum     lesen
   oeffnen
20 rem lesebefehle
30 close 1:rem datei schliessen
```

Achten Sie in den abgebildeten Beispielen bitte vor allem auf die verwendeten Geräteadressen und die Angabe der Übertragungsrichtung (Lesen/Schreiben) durch die Sekundäradresse (Datasette) beziehungsweise die Erweiterung des Dateinamens (Floppy).

Logische Filenummer und – bei Verwendung der Floppy – Sekundäradresse können von Ihnen innerhalb der angegebenen Grenzen beliebig geändert werden. Sollten Sie sich bei der Parameterangabe jedoch unsicher fühlen, können Sie die Beispiele, die sich auf jeden vorkommenden Fall beziehen (Lesen/Schreiben von/auf Floppy/Datasette) völlig unverändert in Ihre eigenen Programme übernehmen.

Lese- und Schreibbefehle

Im Grunde genommen kennen Sie die Befehle zum Lesen und Schreiben von Daten bereits: Es handelt sich um die bekannten Befehle »PRINT» und »INPUT«, allerdings in leicht abgewandelter Form. Wie erwähnt, muß dem C 64 mit Hilfe der logischen Filenummer mitgeteilt werden, welche Datei gemeint ist. Das exakte Format der Lese/Schreibbefehle lautet:

Schreiben: PRINT # (LFN),(DATEN)
und Lesen: INPUT # (LFN),(VARIABLE)

Ebenso wie bei »PRINT« und »INPUT« kann die Aus- beziehungsweise Eingabe mehrerer Daten mit den Zeichen »,« und »;« formatiert werden. Verwenden Sie diese Möglichkeit bitte auf keinen Fall (!), da diese Formatierung nicht einwandfrei funktioniert, wenn sie statt auf den Bildschirm auf Disketten- oder Kassettendateien bezogen wird!

Beispiele:

1. Daten in die Datei »Test« schreiben (Datasette)

```
10 open 1,1,1,"test":rem datei zum
   schreiben oeffnen
20 print#1,"hallo":rem daten schreiben
30 print#1,"nochmal hallo":rem daten schreiben
40 close 1:rem datei schliessen
```

2. Daten aus der Datei »Test« lesen (Datasette)

```
10 open 1,1,0,"test":rem datei zum lesen oeffnen
20 input#1,a$:rem 'hallo' in a$ einlesen
30 input#1,b$:rem 'nochmal hallo' in b$ einlesen
40 close 1:rem datei schliessen
```

3. Daten in die Datei »Test« schreiben (Floppy)

```
10 open 1,8,2,"test,s,w":rem datei zum
   schreiben oeffnen
20 a$="hallo"
30 b$="nochmal hallo"
40 print#1,a$:rem daten schreiben
50 print#1,b$:rem daten schreiben
60 close 1:rem datei schliessen
```

4. Daten aus der Datei »Test« lesen (Floppy)

```
10 open 1,8,2,"test,s,r":rem datei zum     lesen
   oeffnen
20 input#1,a$:rem 'hallo' in a$ einlesen
30 input#1,a$:rem 'nochmal hallo' in b$
einlesen
40 close 1:rem datei schliessen
```


Wie Sie an den unterschiedlichen Beispielen – die Sie übrigens abtippen können, sie sind alle lauffähig – erkennen, ist beim »PRINT # (LFN)«-Befehl ebenso wie beim »PRINT«-Befehl sowohl die unmittelbare Angabe der Daten (in Anführungszeichen) als auch die Angabe von Variablen möglich, deren Inhalt in die Datei geschrieben wird.

Beachten Sie bitte, daß es aufgrund des sequentiellen Datenzugriffs nicht möglich ist, direkt die zweite Zeichenkette »nochmal hallo« einzulesen, ohne zuvor die Zeichenkette »hallo« zu lesen! Der erste »INPUT #«-Befehl liest die erste in die Datei geschriebene Zeichenkette ein, der nächste Befehl die zweite Zeichenkette und so weiter. Der Zugriff beginnt immer am Dateianfang!

Sonderbehandlung leerer Zeichenketten

Ebenso wie manch anderer Spezialfall wird weder im C 64- noch im Floppyhandbuch das Problem leerer Zeichenketten erläutert, die eine spezielle Behandlung erfordern. Dieses Spezialproblem tritt auf, wenn leere Strings, zum Beispiel »a$=" ":print # 1,a$« in eine Datei geschrieben werden sollen, und wird an dieser Stelle auch prinzipiell zuerst einmal übersehen. Auch die in diesem Artikel vorgestellte Schallplattenverwaltung mußte nach eingehendem Testen nachträglich wegen diesem Spezialfall umgeschrieben werden.

Sie werden sich nun fragen, warum es nicht möglich sein soll, eine leere Zeichenkette in eine Datei zu schreiben. Möglich ist es insofern, als der Schreibvorgang problemlos und ohne Fehlermeldung abläuft. Dies ist nicht verwunderlich, da überhaupt nichts, kein einziges Zeichen in die Datei geschrieben wird. Problematisch wird diese Vorgehensweise dadurch, daß in den meisten Fällen nicht von vorn-

herein bekannt ist, ob eine Stringvariable leer ist oder aber einen Inhalt besitzt.

Deutlich wird dies am Beispiel der Schallplattenverwaltung: Jeder Datensatz besteht aus vier Teilfeldern. Wenn Sie nun beispielsweise einen Titel, den Interpreten und die Musikrichtung eingeben, den Titel der Platte jedoch nicht kennen, wird den zugehörigen Strings zum Beispiel folgender Inhalt zugewiesen:

```
t$(2)="Neunundneunzig Luftballons"
i$(2)="Nena"
p$(2)=""
m$(2)="pop"
```

Der String »p$(2)« ist leer. Beim Speichern dieses Datensatzes werden nicht vier, sondern nur drei Strings auf die Diskette beziehungsweise Kassette geschrieben:

```
10 open 1,8,2,"test,s,w" : REM Diskette
20 print#1,t$(2)
30 print#1,i$(2)
40 print#1,p$(2):rem der schreibvorgang entfaellt!
50 print#1,m$(2)
60 close 1
70 open 1,8,2,"test,s,r" : REM Diskette
80 input#1,t$(2)
90 input#1,i$(2)
100 input#1,p$(2):rem einlesen des falschen strings!
110 input#1,m$(2):rem auftreten eines fehlers (blinkende LED)!
120 close 1
```

Mit den ersten beiden Lesebefehlen (Zeile 80 bis 90) werden korrekt die zuvor in »t$(2)« und »i$(2)« enthaltenen und gespeicherten Daten in diese Variablen wieder eingelesen. Der nächste Lesebefehl liest die dritte in der Datei enthaltene Zeichenkette in »p$(2)« ein. Die dritte gespeicherte Zeichenkette entspricht jedoch dem Inhalt von »m$(2)«, da »p$(2)« leer war. Der Variablen »p$(2)« wird daher die Zeichenkette »pop« zugewiesen, der erste Fehler tritt dadurch auf.

Der nächste Fehler ist nicht zu übersehen, da die rote LED der Floppy blinkt. Der Grund: Mit dem vierten »INPUT#«-Befehl (Zeile 110) soll eine weitere Zeichenkette eingelesen werden, obwohl bereits das Ende der Datei erreicht ist, die wie erwähnt nur drei Zeichenketten enthält.

Es ist daher klar, daß das Schreiben leerer Strings in eine Datei unbedingt vermieden werden muß. Eine Möglichkeit besteht darin, einer leeren Stringvariablen vor dem Speichern einen »Scheininhalt« zuzuweisen, zum Beispiel das Zeichen »*« oder ein beliebiges anderes Sonderzeichen.

Immer dann, wenn ein String leer ist, wird das Zeichen »*« in die Datei geschrieben. Beim Einlesen der Datei muß gerade umgekehrt vorgegangen werden: Wenn »*« eingelesen wurde, wird dem betreffenden String der Inhalt » « zugewiesen, er wird als leer definiert. Das entsprechend geänderte Beispielprogramm:

```
10 open 1,8,2,"test,s,w" : REM schreiben
20 if t$(2)="" then t$(2)="*":rem leerer string?
30 print#1,t$(2)
40 if i$(2)="" then i$(2)="*":rem leerer string?
50 print#1,i$(2)
60 if p$(2)="" then p$(2)="*":rem leerer string?
70 print#1,p$(2)
80 if m$(2)="" then m$(2)="*":rem leerer string?
90 print#1,m$(2)
100 close 1

110 open 1,8,2,"test,s,r" : REM lesen
120 input#1,t$(2)
130  if t$(2)="*" then t$(2)="":rem leerer string?
140 input#1,i$(2)
150 if i$(2)="*" then i$(2)="":rem leerer string?
160 input#1,p$(2):rem einlesen des falschen
    strings!
170 if p$(2)="*" then p$(2)="":rem leerer string?
180 input#1,m$(2)
190 if m$(2)="*" then m$(2)="":rem leerer string?
200 close 1
```

Mit dem neu erworbenen Wissen ist es nun möglich, die vorgestellte Schallplattenverwaltung um die Routinen »Datei speichern« und »Datei laden« zu erweitern. Das Prinzip: Beim Speichern der Datei schreiben wir zuerst den Inhalt der Variablen »ad« auf Diskette/Kassette und anschließend in einer Schleife alle Felder des ersten Datensatzes, des zweiten Satzes, bis zuletzt die verschiedenen Teile des letzten Datensatzes mit dem Index »ad« in die Datei geschrieben werden:

```
590 rem  datei speichern
600 if g$="d" then open1,1,1,"schallplatten":
    goto 630:rem datasette
610 open 15,8,15,"s:schallplatten":close 15:
    rem loeschen der alten datei
620 open 1,8,2,"schallplatten,s,w":rem floppy
630 print#1,ad:rem anzahl datensaetze speichern
640 for i=1 to ad
650 if t$(i)="" then t$(i)="*":rem leerer string?
660 print#1,t$(i):rem  t$(1)-t$(ad) speichern
670 if i$(i)="" then i$(i)="*":rem leerer string?
680 print#1,i$(i):rem  i$(1)-i$(ad) speichern
690 if p$(i)="" then p$(i)="*":rem leerer string?
700 print#1,p$(i):rem  p$(1)-p$(ad) speichern
710 if m$(i)="" then m$(i)="*":rem leerer string?
720 print#1,m$(i):rem m$(1)-m$(ad) speichern
730 next
740 close 1:rem datei schliessen
750 return
```

Zeile 600 ist ohne Erläuterung nicht so einfach zu verstehen: Wie wir wissen, müssen unterschiedliche »OPEN«-Befehle verwendet werden, je nach Massenspeicher (Floppy/Datasette). Die vorgestellte Routine zum Speichern der Datei wird beiden Fällen gerecht, wenn das Hauptprogramm um folgende Zeile ergänzt wird:

```
80 g$="f":rem fuer datasette in 'd' ändern
```

In dieser Zeile wird definiert, welches Gerät als Massenspeicher verwendet wird. »d« steht für »Datasette« und »f« für »Floppy«. Ändern Sie diese Zeile in »80 g$ = "d"«, wenn Sie eine Datasette zur Datenspeicherung einsetzen.

Zeile 600 ergibt nun einen Sinn. Sie wird nur dann ausgeführt, wenn die Daten auf Kassette gespeichert werden sollen. Der auf die Datasette zugeschnittene »OPEN«-Befehl wird ausgeführt und die entsprechenden floppyspezifischen Befehle (Zeilen 610 bis 620) werden übersprungen. Nach dem Öffnen der Datei wird die Anzahl der Datensätze auf das Band geschrieben (Zeile 630) und anschließend werden die eigentlichen Datensätze gespeichert (Zeilen 640 bis 730), bevor die Datei geschlossen wird und die Rückkehr zum Hauptprogramm erfolgt. Entsprechend der Sonderbehandlung leerer Zeichenketten werden diese durch das Sonderzeichen »*« ersetzt.

Die Zeile 610, die Sie momentan noch nicht verstehen können, wird in Kürze erläutert. Zuvor will ich jedoch die entsprechende Routine zum Einlesen der Datei vorstellen:

```
770 rem  datei einlesen
780 if g$="d" then open1,1,0,"schallplatten":
    goto 800:rem datasette
790 open 1,8,2,"schallplatten,s,r":rem floppy
800 input#1,ad:rem anzahl datensaetze einlesen
810 for i=1 to ad
820 input#1,t$(i):rem titel einlesen
830 if t$(i)="*" then t$(i)="":rem leerer string?
840 input#1,i$(i):rem interpret einlesen
850 if i$(i)="*" then i$(i)="":rem leerer string?
860 input#1,p$(i):rem platte einlesen
870 if p$(i)="*" then p$(i)="":rem leerer string?
880 input#1,m$(i):rem musikr. einlesen
890if m$(i)="*" then m$(i)="":rem leerer string?
900 next
910 close 1:rem datei schliessen
920 return
```

Wie Sie sehen, verläuft das Einlesen analog zum Speichern: Je nach Gerät wird ein »OPEN«-Befehl zum Lesen von Daten aus der Datei »Schallplatten« von Diskette beziehungs-

weise von Kassette ausgeführt (Zeile 780 bis 790). Danach wird die Anzahl der Datensätze eingelesen (Zeile 800) und anschließend in einer Schleife die Datensätze 1 bis »ad« (Zeile 810 bis 900). Wenn das Sonderzeichen »*« gelesen wird, erhält der betreffende String den Inhalt »«, das heißt er wird als leerer String definiert.

Die Daten müssen beim sequentiellen Zugriff unbedingt (!) in der gleichen Reihenfolge eingelesen werden, wie sie gespeichert wurden, da sonst Fehler auftreten würden.

Das Hauptprogramm muß selbstverständlich um einen Sprung zu dieser Leseroutine erweitert werden. Unmittelbar nach dem Programmstart soll die komplette Datei eingelesen werden, jedoch nur, wenn sie existiert!

Diese Einschränkung ist unbedingt notwendig. Bedenken Sie, daß nach dem erstmaligen Programmstart noch keine Datei auf der Kassette/Diskette vorhanden ist und demzufolge auch nicht eingelesen werden kann. Die einfachste Möglichkeit besteht darin, den Benutzer zu fragen, ob das Programm zum ersten Mal benutzt wird:

```
90 print "'e' druecken, wenn erstbenutzung
   des programms"
100 get a$:if a$="" then 100
110 if a$<>"e" then gosub 780:rem  datei einlesen
```

Durch diese Erweiterung wird der Benutzer nach dem Programmstart aufgefordert, die Taste »e« zu drücken, wenn es sich um die Erstbenutzung des Programms handelt. Drückt der Benutzer hingegen eine beliebige andere Taste, existiert somit bereits eine Datei, wird die Leseroutine gestartet und die Datei eingelesen.

Der Befehlskanal

Die Zeile 620, in der die Floppydatei eröffnet wird, müßte bei aufmerksamer Lektüre des bisher Gesagten problemlos zu verstehen sein. Zeile 610 jedoch stellt Sie zweifellos wieder vor schwierige Probleme. In dieser Zeile wird der sogenannte »Befehlskanal« zur Floppy eröffnet. Dieser Kanal dient nicht zum Ansprechen einer Datei, sondern zur Übermittlung spezieller Befehle an die Floppy. Der jeweilige Befehl wird ebenso in Anführungszeichen angegeben, wie es bei der Angabe eines Dateinamens der Fall ist. Der verwendete Befehl »s:schallplatten« führt dazu, daß eine eventuell auf der Diskette bereits unter dem Namen »Schallplatten« vorhandene Datei gelöscht wird.

Der Grund: Die Datasette bemerkt nicht, ob durch das Schreiben von Daten auf das Band womöglich eine Datei überschrieben wird, die ebenfalls an jener Bandposition aufgezeichnet wurde. Das Überschreiben einer alten Schallplattendatei durch eine eventuell geänderte, in der Datensätze zum Beispiel gelöscht oder neu eingetragen wurden, geschieht daher automatisch, wenn Sie das Band vor Anwahl des Kommandos »Ende« im Menü immer zur gleichen Position zurückspulen.



Die Floppy VC 1541 ist im Gegensatz zur Datasette ein »hochintelligentes« Gerät, das sich weigert, eine bereits existierende Datei zu überschreiben. Ein entsprechender Versuch führt zum Blinken der roten LED am Diskettenlaufwerk, durch das ein Fehler angezeigt wird. Diese Einrichtung ist zwar prinzipiell sehr sinnvoll, um ein versehentliches Überschreiben zu verhindern, in unserem Fall soll dieses Überschreiben der alten durch die neue Datei jedoch bewußt vorgenommen werden.

Die einfachste Lösung besteht darin, die alte Datei zu löschen, bevor die neue Datei gespeichert wird. Dazu wird der Befehlskanal verwendet, über den der Löschbefehl an die Floppy übermittelt wird. Beachten Sie bitte, daß der Befehlskanal nach Übermittlung des Befehls wieder geschlossen werden muß (»CLOSE 15«). Das allgemeine Format dieses Löschbefehls lautet:

OPEN 15,8,15,"S:(DN)"

»S« bedeutet »Scratch« (=Löschen) und »DN« ist der Name der zu löschenden Datei. Die Befehlsübermittlung ist Ihnen bereits bekannt, da Sie – wenn Sie eine Floppy besitzen – mit Sicherheit bereits mehrere Disketten mit dem Befehl »OPEN 15,8,15,"N:(NAME,ID)"« formatiert, das heißt zur Benutzung vorbereitet haben.

Was Ihnen jedoch nicht unbedingt bekannt ist: Über den Befehlskanal kann zusätzlich beim Auftreten eines Fehlers (Blinken der roten LED) ermittelt werden, um welchen Fehler es sich handelt. Verwenden Sie hierzu bitte folgendes Programm:

```
10000 rem fehlermeldung lesen
10010 open 15,8,15:rem befehlskanal oeffnen
10020 input#15,a$,b$,c$,d$:rem fehlermeldung lesen
10030 close 15:rem befehlskanal schliessen
10040 print a$;b$,c$;d$:rem fehlermeldung ausgeben
```

Dieses Programm können Sie jederzeit eingeben, wenn ein Programm statt zum einwandfreien Lesen oder Schreiben von Daten zum Blinken der LED führt. Ich habe bewußt hohe Zeilennummern gewählt, um ein teilweises Überschreiben Ihres eigenen Programms zu verhindern. Starten Sie das Programm nach der Eingabe mit »RUN 10000«.

Auf dem Bildschirm wird die Fehlermeldung im Klartext ausgegeben. Zusätzlich erhalten Sie die Nummer des aufgetretenen Fehlers (siehe Floppy-Handbuch) und die Angabe einer Spur und eines Sektors. Diese Angaben werden Ihnen momentan noch nichts nützen, sie sind jedoch nur in den seltensten Fällen notwendig zur Fehlersuche. Sie geben die genaue Position auf der Diskette an, an der der Fehler auftrat.

Die häufigsten Fehler

Seien Sie versichert, daß Sie bei der Arbeit mit sequentiellen Dateien immer wieder das Blinken der roten LED zu sehen bekommen, ein Anblick, an den Sie sich schon bald gewöhnen werden. Wenn Sie daraufhin jedesmal wie beschrieben den Fehler einlesen, werden Sie feststellen, daß von den Dutzenden möglichen häufig die gleichen Fehler auftreten:

1. »Write protect on«: Sie versuchten trotz Schreibschutz Daten auf die Diskette zu schreiben. Entfernen Sie also den kleinen Klebestreifen von der Diskette.

2. »Drive not ready«: Es ist empfehlenswert, beim nächsten Schreib- oder Leseversuch eine Diskette einzulegen.

3. »File exists«: Dieser Fehler ist nicht ganz so trivial wie die zuvor erwähnten. Sie versuchten, ein Programm zu speichern oder aber Daten in eine Datei zu schreiben, obwohl ein Programm oder eine Datei mit dem betreffenden Namen bereits auf der Diskette existiert. Bedenken Sie, daß die Floppy ein Überschreiben nicht zuläßt und löschen Sie das entsprechende Programm beziehungsweise fügen Sie wie

beschrieben den Befehl zum Löschen der betreffenden Datei in Ihr Programm ein.

4. »File data error in (Zeilennummer)«: Diese Fehlermeldung tritt bei der Arbeit mit Dateien zweifellos am häufigsten auf. Obwohl in den Beispielprogrammen bisher nur Stringvariablen verwendet wurden, werden Sie schnell herausfinden, daß auch numerische Variablen in eine Datei geschrieben und gelesen werden können. Diese »Vermischung« numerischer und Stringvariablen bietet jedoch ein Risiko: Wenn Sie zum Beispiel mit dem Befehl »INPUT #1,A« eine numerische Variable einlesen wollen, sich an dieser Stelle der Datei jedoch eine Stringvariable befindet, wird ein »File data error« gemeldet, wie im folgenden Beispiel:

```
10 open 1,8,1,"test,s,w":rem diskettendatei zum
   schreiben oeffnen
20 a$="dies ist ein test"
30 a=10
40 print#1,a$:rem string schreiben
50 print#1,a:rem zahl schreiben
60 close 1:rem datei schliessen
70 open 1,8,1,"test,s,r":rem diskettendatei zum lesen oeffnen
80 input#1,a:rem numerische variable lesen
90 input#1,a$:rem stringvariable lesen
100 close 1:rem datei schliessen
```

Haben Sie den Fehler bemerkt?

Nach dem Starten dieses Programms wird ein »File data error in 80« gemeldet, da zuerst eine Stringvariable, anschließend eine numerische Variable in die Datei geschrieben wurde. Nun jedoch wird zuerst eine numerische Variable eingelesen, obwohl sich am Dateianfang die Zeichenkette »dies ist ein test« befindet. Sie werden verstehen, daß es dem C 64 nicht möglich ist, diese Zeichenkette der numerischen Variablen »a« zuzuweisen.

5. »String to long error in (Zeilennummer)«: Auch dieser Fehler tritt sehr häufig auf. Die Ursache wird in zumindest einigen Exemplaren des C 64-Handbuches totgeschwiegen: Der »INPUT # (LFN)«-Befehl kann nur Zeichenketten mit einer maximalen Länge von 80 Zeichen einlesen, die Zeichenkette, die in der angegebenen Zeile gelesen werden sollte, ist jedoch länger. Beispiel:

```
10 open 1,8,1,"test,s,w"
20 a$="dies sind 105 zeichen":rem 21 zeichen laenge
30 b$=a$+a$+a$+a$+a$:rem 105 zeichen laenge
40 print#1,b$:rem string mit 105 zeichen schreiben
50 close 1
60 open 1,8,1,"test,s,r"
70input#1,b$:rem hier tritt der fehler auf!
80 close 1
```

Die unangenehmsten Fehler sind die sogenannten »unsystematischen Fehler«, die nicht regelmäßig, sondern nur manchmal auftreten. Jeder Programmierer fürchtet diese Fehlerart, da das unsystematische Auftreten die Fehlersuche enorm erschwert.

Überprüfen Sie beim Auftreten unsystematischer Fehler den Sonderfall der leeren Zeichenketten, der häufig die Ursache darstellt. Wenn Sie diesen Fall nicht durch den Einsatz eines Sonderzeichens behandeln, vergewissern Sie sich bitte, ob es völlig ausgeschlossen ist, daß in bestimmten Fällen in Ihrem Programm leere Zeichenketten in eine Datei geschrieben und wieder gelesen werden sollen!

Nicht unbedingt zu einer echten Fehlermeldung, aber immer zum Einlesen falscher Daten führt das nicht im Handbuch erläuterte Schreiben »verbotener« Zeichen in eine Datei. Diese »verbotenen« Zeichen sind das Anführungszeichen, das Komma und das Semikolon (",;). Achten Sie bitte darauf, daß in den Zeichenketten, die Sie in eine Datei schreiben, keinesfalls (!) eines dieser Zeichen vorkommt. Das heißt, verzichten Sie in der Schallplattenverwaltung auf diese Zeichen bei der Eingabe von Datensätzen!

Wir sind nun am Ende dieser Einführung angelangt, die es Ihnen ermöglichen sollte, eigene einfache Dateiverwaltungen zu programmieren. Eine Bemerkung zum Schluß: Über die Möglichkeiten, die eine Dateiverwaltung bietet, den Suchkomfort, die Geschwindigkeit und andere wesentliche Leistungsmerkmale entscheiden vor allem die verwendeten »Datenstrukturen«, das heißt die Art und Weise, wie die Daten miteinander verknüpft werden und die Datenorganisation, die Art der Anordnung der Daten im Computerspeicher und auf dem Speichermedium. In diesem Artikel wurden die einfachsten Datenstrukturen, sogenannte »einfache sequentielle Listen« und die einfachst mögliche Datenorganisation, die »sequentielle Organisation« besprochen.

Mit diesen Strukturen lassen sich voll funktionsfähige Dateiverwaltungen aufbauen. Sollten Sie jedoch Geschmack an der Dateiverwaltung finden, werden Sie nicht umhin kommen, sich auch mit weitaus komplexeren Strukturen wie zum Beispiel »verketteten« Listen oder »Baumstrukturen« zu befassen und mit den Verfahren zur Indizierung und Sortierung von Dateien zu beschäftigen. Sollten Sie an diesen – zugegebenermaßen nicht ganz einfachen – Themen Interesse finden, empfehle ich Ihnen mein Buch »Alles über Dateiverwaltung und Datenbanken«, in dem diese anspruchsvolleren Datenstrukturen und Möglichkeiten zur Datenorganisation umfassend erläutert werden.

Zum Abschluß stelle ich Ihnen das komplette, abtippfertige Listing des von mir verwendeten Demoprogramms vor.

(S. Baloui/tr)

Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel {CLR}, muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksummer 64 auf Seite 135.

```
10 REM --- SCHALLPLATTENVERWALTUNG ---        <147>
20 REM ----- (S.BALOUI/FEB 1986) -----        <202>
30 :                                          <006>
40 REM --- DIMENSIONIERUNG ---                <003>
50 DIM T$(500),I$(500),P$(500),M$(500)        <236>
60 :                                          <036>
70 REM  --- VORBEREITUNG ---                  <036>
80 G$="F":REM  BEI FLOPPYBETRIEB IN 'F' AE
   NDERN                                      <203>
90 PRINT "'E' DRUECKEN, WENN ERSTBENUTZUNG
    DES PROGRAMMS"                            <024>
100 GET A$:IF A$="" THEN 100                  <229>
110 IF A$<>"E" THEN GOSUB 780:REM  DATEI E
    INLESEN                                   <243>
120 :                                         <096>
130 REM --- MENUE ---                         <130>
140 PRINT CHR$(147):REM SCREEN LOESCHEN       <082>
150 PRINT "(E)INTRAGEN VON DATENSAETZEN"      <010>
160 PRINT "(S)UCHEN VON DATENSAETZEN"         <202>
170 PRINT "(B)ENDEN DES PROGRAMMS             <116>
180 PRINT:PRINT "KOMMANDO ?"                  <172>
190 GET A$:IF A$="" THEN 190                  <129>
200 IF A$="E" THEN GOSUB 260                  <099>
210 IF A$="S" THEN GOSUB 340                  <092>
220 IF A$="B" THEN GOSUB 600:END              <132>
230 GOTO 140                                  <222>
240 :                                         <216>
250 REM --- EINTRAGEN ---                     <034>
260 AD=AD+1:REM ANZAHL DATENSAETZE UM EINS
     ERHOEHEN                                 <083>
270 INPUT "TITEL";T$(AD)                      <092>
280 INPUT "INTERPRET";I$(AD)                  <226>
290 INPUT "PLATTE";P$(AD)                     <091>
300 INPUT "MUSIKRICHTUNG";M$(AD)              <198>
310 RETURN                                    <114>
320 :                                         <042>
330 REM --- SUCHEN ---                        <004>
340 INPUT "SUCHKRITERIUM";S$                  <249>
350 FOR I=1 TO AD                             <014>
```

Listing. Die komplette, im Kurs erarbeitete Schallplattenverwaltung

```
360 IF S$<>T$(I) AND S$<>I$(I) AND S$<>P$(
    I) AND S$<>M$(I) THEN 460              <175>
370 PRINT T$(I):PRINT I$(I):PRINT P$(I):PR
    INT M$(I)                              <156>
380 PRINT "WEITERSUCHEN (J/N) ?"           <014>
390 GET A$:IF A$="" THEN 390               <139>
400 IF A$="J" THEN 460                     <004>
410 PRINT "(L)OESCHEN/(A)ENDERN ?"         <137>
420 GET A$:IF A$="" THEN 420               <007>
430 IF A$="L" OR A$="A"THEN N=I:GOSUB 500:
    REM LOESCHEN                           <023>
440 IF A$="A" THEN GOSUB 260:REM EINTRAGEN <104>
450 GOTO 470:REM ENDE                      <117>
460 NEXT                                   <216>
470 RETURN                                 <018>
480 :                                      <202>
490 REM --- LOESCHEN ---                   <041>
500 FOR I=N TO AD-1                        <098>
510 T$(I)=T$(I+1)                          <096>
520 I$(I)=I$(I+1)                          <053>
530 P$(I)=P$(I+1)                          <098>
540 M$(I)=M$(I+1)                          <093>
550 NEXT                                   <052>
560 AD=AD-1:REM ANZAHL DATENSAETZE UM EINS
    ERNIEDRIGEN                            <205>
570 RETURN                                 <120>
580 :                                      <048>
590 REM --- DATEI ABSPEICHERN ---          <136>
600 IF G$="D" THEN OPEN 1,1,1,"SCHALLPLATT
    EN":GOTO 630:REM DATASETTE             <015>
610 OPEN 15,8,15,"S:SCHALLPLATTEN":CLOSE 1
    5:REM LOESCHEN DER ALTEN DATEI         <119>
620 OPEN 1,8,2,"SCHALLPLATTEN,S,W":REM FLO
    PPY                                    <163>
630 PRINT#1,AD:REM ANZAHL DATENSAETZE SPEI
    CHERN                                  <121>
640 FOR I=1 TO AD                          <050>
650 IF T$(I)="" THEN T$(I)="*":REM LEERER
    STRING?                                <160>
660 PRINT#1,T$(I):REM  T$(1)-T$(AD) SPEICH
    ERN                                    <241>
670 IF I$(I)="" THEN I$(I)="*":REM LEERER
    STRING?                                <171>
680 PRINT#1,I$(I):REM  I$(1)-I$(AD) SPEICH
    ERN                                    <199>
690 IF P$(I)="" THEN P$(I)="*":REM LEERER
    STRING?                                <104>
700 PRINT#1,P$(I):REM  P$(1)-P$(AD) SPEICH
    ERN                                    <073>
710 IF M$(I)="" THEN M$(I)="*":REM LEERER
    STRING?                                <051>
720 PRINT#1,M$(I):REM M$(1)-M$(AD) SPEICHE
    RN                                     <191>
730 NEXT                                   <232>
740 CLOSE 1:REM DATEI SCHLIESSEN           <015>
750 RETURN                                 <044>
760 :                                      <228>
770 REM --- DATEI EINLESEN ---             <012>
780 IF G$="D" THEN OPEN 1,1,0,"SCHALLPLATT
    EN":GOTO 800:REM DATASETTE             <053>
790 OPEN 1,8,2,"SCHALLPLATTEN,S,R":REM FLO
    PPY                                    <059>
800 INPUT#1,AD:REM  ANZAHL DATENSAETZE EIN
    LESEN                                  <117>
810 FOR I=1 TO AD                          <222>
820 INPUT#1,T$(I):REM  TITEL EINLESEN      <255>
830 IF T$(I)="*" THEN T$(I)="":REM LEERER
    STRING?                                <003>
840 INPUT#1,I$(I):REM  INTERPRET EINLESEN  <000>
850 IF I$(I)="*" THEN I$(I)="":REM LEERER
    STRING?                                <094>
860 INPUT#1,P$(I):REM  PLATTE EINLESEN     <114>
870 IF P$(I)="*" THEN P$(I)="":REM LEERER
    STRING?                                <139>
880 INPUT#1,M$(I):REM  MUSIKR. EINLESEN    <135>
890 IF M$(I)="*" THEN M$(I)="":REM LEERER
    STRING?                                <038>
900 NEXT                                   <148>
910 CLOSE 1:REM DATEI SCHLIESSEN           <187>
920 RETURN                                 <216>
```

Listing. Schallplattenverwaltung (Schluß)

Das DOS 5.1 auf der Demo-Diskette

Einsteiger wissen meist gar nicht, was sie mit dem Programm »DOS 5.1« auf ihrer Demo-Diskette alles anstellen können. Hier erfahren Sie, wie Sie dieses Hilfsprogramm, das den Umgang mit Ihrer Floppy erleichtert, bedienen können.

Mit jedem Diskettenlaufwerk 1541 wird eine Demo-Diskette mitgeliefert. Auf dieser Diskette sind einige Programme, von denen der von jeglichem Computerwissen unberührte Neuling nicht genau weiß, wie er sie einzusetzen hat. Unter anderem ist das das Programm »DOS 5.1«. Dieses Programm erlaubt die bequeme Nutzung vieler Diskettenbefehle.

Was macht das DOS 5.1?

Es liefert drei verschiedene Arten von Befehlen:
- Der Fehler-Status kann mit einem einfachen Befehl gelesen werden;
- das Inhaltsverzeichnis (Directory) einer beliebigen Diskette kann gelesen werden, ohne ein im Speicher befindliches Programm zu zerstören;
- eine Anzahl von Diskettenbefehlen kann einfach durchgeführt werden.

```
    **** COMMODORE 64 BASIC V2 ****
 64K RAM SYSTEM  38911 BASIC BYTES FREE
READY.
LOAD"C-64 WEDGE",8
SEARCHING FOR C-64 WEDGE
LOADING
READY.
RUN

      DOS MANAGER V5.1/071382
          BY  BOB FAIRBAIRN
(C) 1982 COMMODORE BUSINESS MACHINES
READY.
```

Bild 1. So meldet sich das DOS 5.1

Wie man das DOS 5.1 in den Computer lädt

Vielleicht haben Sie es schon einmal mit
LOAD"DOS 5.1",8
probiert und haben dann lediglich einen Syntax-Error erhalten. Das liegt daran, daß das DOS ein Maschinenprogramm ist. Es gibt auf der Diskette aber ein anderes Programm, das sich »C-64 Wedge« nennt. Und dieses Programm ist das

Ladeprogramm für das DOS 5.1. Also laden Sie es mit
LOAD"C-64 WEDGE",8
und starten es durch RUN. Dadurch wird das DOS automatisch in den Speicher des Computers geladen. Nach kurzer Zeit meldet sich der Computer mit einer kurzen Einschaltmeldung (Bild 1).

Jetzt können Sie das DOS nutzen. Es macht nichts, wenn Sie NEW eintippen: Das DOS 5.1 liegt in einem Speicherbereich, der mit diesem Befehl nicht gelöscht werden kann. Sie können auch ruhig ein anderes Basic-Programm laden und starten. Es ist ohne weiteres möglich, und die beiden Programme beeinflussen einander nicht.

Die Befehle des DOS 5.1

Die meisten DOS-Befehle beginnen mit einem »>« oder »@«. Man kann sie wahlweise benutzen, sie bewirken das gleiche. Da jedoch das »@« mit einer Taste gedrückt werden kann, wird im folgenden nur dieses Zeichen verwendet. Doch dazu später. Es gibt noch andere Zeichen, mit denen DOS-Befehle beginnen können. Diese Zeichen (»←«, »↑« und »/«) haben folgende Bedeutung:

»←«: Speichern (SAVE) eines Programms. Beispiel: ←TEST speichert ein Programm namens TEST auf die Diskette. Sonst würde man
SAVE"TEST",8
schreiben.

»↑«: lädt ein Programm (LOAD) von Diskette und startet es automatisch. Beispiel: ↑TEST lädt das Programm TEST von der Diskette und startet es automatisch. Hiermit wird der Befehl
LOAD"TEST",8:RUN
ersetzt.

»/«: lädt ein Programm, ohne es zu starten. Beispiel: /TEST entspricht dem Befehl
LOAD"TEST",8

In vielen Fällen ist es nicht nötig, beim Laden eines Programms den Programmnamen voll auszuschreiben. Man kann Teile des Namens eingeben und »?« beziehungsweise »*« für den Rest eingeben.

Beispiel: Der Befehl /TE* lädt das erste Programm von der Diskette, dessen erste beiden Buchstaben T und E sind. Das kann das Programm TE sein, aber auch TEST, TERMIN oder TEUER, falls diese Programme auf der Disk vorhanden sind. Der Befehl /:* lädt das erste Programm auf der Diskette.

Das »?« ersetzt irgendein Zeichen im Programmnamen: /T??T lädt jedes Programm, dessen erster und vierter Buchstabe ein T ist. Das könnte ein Programm namens TEST, TAST oder TILT sein.

Benutzen Sie »?« und »*« aber nicht für den SAVE-Befehl (beziehungsweise ←). Beim Speichern muß der Programmname natürlich voll ausgeschrieben werden.

Disk-Status

Dies ist der kürzeste Befehl von DOS 5.1. Sie brauchen lediglich das @-Symbol einzugeben, gefolgt von der RETURN-Taste, und Sie erhalten den Floppy-Status angezeigt. Wenn die Floppy keinen Fehler meldet, so erhalten Sie die Meldung 00,OK,00,00. Falls Sie ein oder mehrere Programme mit dem SCRATCH-Befehl gelöscht haben, erhalten Sie eine Information über die Anzahl der von der Diskette gelöschten Programme. Wenn die Diskettenstation durch Blinken der roten Lampe einen Fehler anzeigt, gibt Ihnen dieser Befehl die Art des Fehlers an. Falls Sie zum Beispiel ein Programm von der Diskette laden wollen, das nicht existiert, so erhalten Sie nach Eingabe dieses Befehls die Meldung 62,FILE NOT FOUND,00,00. Manchmal reicht eine solche



Meldung aber zum Verständnis nicht aus. Dann sollten Sie im Floppy-Handbuch nachlesen. Wollten Sie diese Meldung ohne DOS erhalten, müßten Sie normalerweise dieses kleine Programm eingeben:

```
10 OPEN 15,8,15
20 INPUT#15,A1,A2$,A3,A4
30 PRINT A1,A2$,A3,A4
40 CLOSE 15
```

Directory

Sie können das Directory einer Diskette lesen, ohne das im Computer befindliche Programm zu zerstören. Tippen Sie einfach ein: @$ und natürlich die Return-Taste. Sie können auch eine bestimmte Auswahl der anzuzeigenden Programme treffen, indem Sie die Abkürzungen »?« und »*« verwenden. Geben Sie beispielsweise ein: @$:D* und Sie erhalten jedes Programm, das mit einem D beginnt. Die Eingabe von @$:??? listet jedes aus drei Buchstaben bestehende Programm.

New (Formatieren)

Um eine Diskette neu zu formatieren, müßten Sie ohne DOS folgendes eingeben:
OPEN 15,8,15,"N:1541 TEST/DEMO,XY":CLOSE 15
Mit DOS 5.1 geben Sie nur ein:
@ N:1541 TEST/DEMO,XY

Der Befehl @N:1541 TEST/DEMO ohne Angabe der ID-Nummer (in unserem Fall XY) formatiert eine Diskette nicht neu. Er gibt der Diskette lediglich einen neuen Namen und löscht dabei das ganze Directory. Diese Befehlsvariante läuft wesentlich schneller ab als der komplette N-Befehl. Allerdings muß die Diskette vorher schon formatiert gewesen sein.

Initialise

Der Befehl dazu lautet: @I

Er wird normalerweise nicht benötigt, kann jedoch manchmal helfen, wenn Sie zum Beispiel ein DRIVE NOT READY erhalten. Dieser Befehl sagt der Floppy, daß die Kenndaten der momentan im Laufwerk befindlichen Diskette noch einmal in den Puffer (Zwischenspeicher) der Floppy eingelesen und das Laufwerk quasi einen Neustart machen soll. Dies kann auch erforderlich sein, falls sich der Schreib-/Lesekopf auf einer undefinierten Spur »festgebissen« hat und nicht mehr zurückfindet.

Scratch

Sollen Programme gelöscht werden, so ist einzugeben: @S:Programmname

Der Befehl @S:D* löscht alle Programme von der Diskette, die mit D beginnen. Mit @S:* wird die gesamte Diskette gelöscht, ähnlich dem N-Befehl, ist jedoch langsamer in der Ausführung.

Rename

Um den Namen eines Programmes auf der Diskette zu ändern, tippen Sie ein:

@ R:neuer Name=alter Name

Beispiel: Aus dem Programm TEST wird das Programm Lampe, wenn Sie eingeben:

@ R:LAMPE=TEST

Copy

In der Regel wird der Copy-Befehl nur bei Doppellaufwerken benutzt. Er ist jedoch auch für Einzellaufwerke anwendbar. Dieser Befehl verdoppelt ein Programm oder eine Datei auf der Diskette. @ C:HUND=KATZE erstellt eine Kopie der Datei KATZE mit dem Namen HUND.

Eine weniger bekannte Möglichkeit des Copy-Befehls ist die, daß zwei verschiedene Dateien miteinander zu einer einzigen Datei verbunden werden können. Das kann so gemacht werden:

@ C:STREIT=HUND,KATZE

Die Dateien HUND und KATZE werden zu der neuen Datei STREIT zusammengefügt.

Geheimnisvolles DOS

Doch das DOS 5.1 hat noch einige weitere nützliche Befehle:

- %NAME bedeutet, ein Maschinenprogramm absolut zu laden. (Dabei kann es bei normalem Betrieb gelegentlich vorkommen, daß der Computer ein OUT OF MEMORY ausgibt – unter DOS 5.1 wird dies vermieden.)
- @#9 lenkt das DOS auf die Floppy mit der Geräteadresse 9. (Es ist auch möglich, durch Angabe der Geräteadresse 1 die Funktionen auf die Datasette zu lenken. Hierbei können aber nur die Befehle /, ↑ und % benutzt werden. Der SAVE-Befehl muß ausgeschrieben werden.)
- @Q schaltet das DOS ab. (Ein Neustart ist durch Eingabe von SYS 52224 möglich.)
- @$:*=PRG listet nur PRG-Files. Entsprechendes gilt für @$:TE*=SEQ, @$:?E=REL und @$:T??T=USR

Ein paar zusätzliche Tips

Die Ausgabe des Directories mit @$ läßt sich über die Space-Taste steuern. Man spart sich Tipparbeit, wenn man sich mit @$ das Inhaltsverzeichnis auf den Bildschirm holt. Danach fährt man mit dem Cursor hoch zum gewünschten Programmnamen und drückt die Tasten »/«, »↑«, »%« oder »←« und ein Return. Schon erscheint die Meldung LOADING beziehungsweise SAVING des Programms.

Wollen Sie die DOS-Befehle auch in Programmen verwenden, so müssen die Namen der Befehle in Anführungszeichen eingegeben werden. Die Befehle lauten dann zum Beispiel »@"$"«, »/"TEST"« oder »←"TEST"«.

Stört Sie die Tatsache, daß das DOS auf normalem Wege (also LOAD und SAVE) nicht kopierbar ist? Machen Sie aus ihm ein normal kopierbares Basicprogramm. Laden Sie bitte das DOS auf normalem Weg, falls es ohnehin nicht schon geschehen ist. Tippen Sie nun Listing 1 (DOS verschieben) ein. Nach einem RUN ohne DATAfehler stehen nun beide hintereinander im Speicher.

Löschen Sie jetzt alle Programmzeilen außer Zeile 100. Es darf kein NEW eingegeben werden. Es können aber beliebig viele Zeilen hinzugefügt werden. Zum Beispiel könnte man die Bildschirmfarben festlegen und alle Tasten mit Wiederholfunktion belegen. In diesem Fall können Sie noch das Listing 2 zusätzlich eingeben.

Damit ist Ihre erweiterte DOS-Version fertig und kann auf normalem Wege gespeichert werden.

(Herbert Heise/gk/dm)

Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel {CLR}, muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksummer 64 auf Seite 135.

```
2 REM DOS VERSCHIEBEN 11.05.1984                   <214>
3 REM BY HERBERT HEISE                             <142>
10 IF PEEK(46)>12 THEN PRINT"{CLR}DAS PROG
   RAMM SOLLTE NUR EINMAL GESTARTET{2SPACE
   }WERDEN!":END                                   <071>
20 GOSUB 310:POKE 46,PEEK(46)+4                    <246>
30 POKE 45,PEEK(45)+29 AND 255                     <061>
40 IF PEEK(45)<29 THEN POKE 46,PEEK(46)+1          <245>
50 CLR:A=PEEK(45)+PEEK(46)*256-1024-29             <046>
60 B=52224                                         <076>
70 FOR I=0 TO 1023:POKE A+I,PEEK(B+I):NEXT         <199>
80 A=A+1024                                        <081>
90 FOR I=0 TO 28:READ B:POKE A+I,B:NEXT            <214>
100 A=PEEK(45)+PEEK(46)*256-29:SYS A:END           <248>
310 FOR I=0 TO 28:READ B:S=S+B:NEXT                <099>
320 IF S<>3210 THEN PRINT"DATAFEHLER!":END         <203>
330 PRINT"DATA OK!":RESTORE:RETURN                 <103>
1000 DATA 164,20,165,21,132,90,133,91,56           <217>
1010 DATA 233,4,132,95,133,96,160,0,169            <036>
1020 DATA 208,132,88,133,89,32,191,163             <174>
1030 DATA 76,0,204                                 <065>
```

Listing 1. Hiermit wird das DOS kopierbar wie ein Basic-Programm.

```
2 REM NEUE FARBEN                                  <032>
3 REM BY HERBERT HEISE                             <142>
10 POKE 53280,6:REM RAHMEN                         <146>
20 POKE 53281,6:REM HINTERGRUND                    <103>
30 POKE 646,1:REM SCHRIFT                          <048>
40 POKE 650,128:REM REPEAT-FUNKTION                <197>
100 A=PEEK(45)+PEEK(46)*256-29:SYS A:END           <248>
```

Listing 2. Festlegen der Bildschirmfarben beim Einschalten.

Drucken ohne Rätsel



Wenn man sich ein wenig intensiver mit seinem Computer beschäftigt, so ist die Anschaffung eines Druckers in der Regel unvermeidlich. Es handelt sich dabei nicht etwa um Luxus, sondern vielmehr um ein sehr hilfreiches Zusatzgerät, das einem die Arbeit stark erleichtert. Es stellt sich dabei natürlich vielfach die Frage, wie ein solcher Drucker prinzipiell angesprochen oder bedient wird.

Wir wollen Ihnen nun zeigen, wie die gängigsten Drucker für den C64 angesprochen werden. Es soll erläutert werden, welche Möglichkeiten ein Drucker bietet und wie diese Möglichkeiten genutzt werden können. Bei den Druckern für den Commodore 64 und anderen Heim- oder Personal-Computern handelt es sich in der Regel um sogenannte Matrixdrucker. Das heißt nichts anderes, als daß der Drucker die einzelnen Zeichen aus winzigen Punkten zusammensetzt, wobei jeder Punkt durch eine Nadel, die gegen ein Farbband auf das Papier schlägt, gedruckt wird. Genauere Angaben zur Funktionsweise eines solchen Matrixdruckers finden Sie im Artikel »Ohne Drucker geht es nicht – eine Entscheidungshilfe«.

Es stellt sich nun natürlich die Frage, wie ein Drucker die Informationen verarbeitet, die vom Computer kommen und vor allem, wie der Benutzer einen Ausdruck herstellen kann. Um das genauer zu verstehen, müssen wir erst einmal begreifen, wie ein Drucker im Prinzip aufgebaut ist. Bei einem der gängigen Matrixdrucker handelt es sich in der Regel um nichts anderes, als um einen vollständigen kleinen Computer, der so wie der C64 ein eigenes ROM, ein RAM und einen Prozessor besitzt. Zusätzlich ist natürlich noch eine Elektronik zur Steuerung des Druckvorgangs vorhanden. Das ROM des Druckers, also der Festwertspeicher, enthält dabei ein Betriebssystem und die Punktmatrix für jedes einzelne Zeichen, das gedruckt werden kann. Das RAM dient dem Drucker vor allem als Zwischen- und Pufferspeicher. Es erlaubt dem Computer eine effektivere Drucksteuerung. Wenn zum Beispiel ein Text ausgedruckt werden soll, so schickt der Computer die Zeichen an den Drucker. Dieser füllt seinen Pufferspeicher, und während er dann mit dem Drucken des Speicherinhalts beschäftigt ist, kann der Computer wieder andere Arbeiten übernehmen. Der Prozessor des Druckers steuert, wie im Computer auch, sämtliche Abläufe im Drucker. Haben Sie also nun einen Drucker an Ihren Computer angeschlossen, so stellt sich beim Zugreifen auf den Drucker zuerst die Frage, was der Drucker mit den

Daten, die Sie ihm schicken, anfangen soll. Dazu muß man wissen, daß ein Drucker normalerweise über zwei verschiedene Betriebsarten verfügt und zwar über eine Betriebsart, in der alle entgegengenommenen Zeichen direkt ausgedruckt werden und eine andere Betriebsart, in der ein Zeichen als Befehl interpretiert wird. Ja, Sie haben ganz richtig gelesen: Ihr Drucker kann verschiedene Befehle ausführen. Zu diesem Zweck hat er einen bestimmten Befehlssatz, der von Druckertyp zu Druckertyp unterschiedlich sein kann; jedoch in wesentlichen Punkten bei den verbreiteten Typen übereinstimmt. Für uns sind im wesentlichen zwei dieser Befehlsformate von Bedeutung. Das ist zum einen der Befehlssatz der Commodore-Drucker oder der Drucker mit Commodore-Interface und zum anderen der Befehlssatz der Epson-Drucker. Der Epson-Befehlssatz ist dabei unter dem ESC/P-Standard (Epson Standard Code for Printers) zusammengefaßt, an den sich die meisten der bekannten Druckertypen halten. Diese Druckertypen sind dabei, wie die Epson-Drucker, mit einer Centronics kompatiblen Schnittstelle ausgestattet und lassen sich an einen C 64 nicht direkt anschließen. Der Anschluß erfolgt hier mit speziellen Interfaces oder über den User-Port mit einer Software-Centronics-Schnittstelle. Dazu kommen wir später. Zuerst sollen uns die Commodore-Drucker interessieren.

Es stellt sich an dieser Stelle natürlich die Frage, was für einen Sinn die einzelnen Befehle bei den Druckern haben. Diese Frage ist schnell beantwortet: Die Befehle dienen der Wahl der jeweiligen Druckmodi. Wenn Sie ein Zeichen auf dem Bildschirm Ihres Computers ausgeben, so wird das Zeichen so dargestellt, wie der Bildschirm das zuläßt; also in einem bestimmten Punktraster mit einer bestimmten Größe und so weiter – bei einem Drucker ist das anders. Hier gibt es in der Regel sehr verschiedene Möglichkeiten, ein Zeichen zu drucken. Es kann dick, dünn, hoch, niedrig, normal oder invers sein. Darüber hinaus kann der Zeilenabstand zwischen zwei Zeilen eingestellt werden. Bei einem Drucker ist es möglich, die folgende Zeile direkt an die vorhergehende anzuschließen oder einen Zwischenraum zur besseren Lesbarkeit zu lassen.

Alle diese Einstellungen müssen einem Drucker natürlich mitgeteilt werden. Er muß also wissen, ob er als nächstes einen Befehl ausführen oder ein Zeichen drucken soll. Die Commodore-Drucker lehnen sich dabei verständlicherweise an den Zeichensatz des Computers an. Dieser kann auf dem Bildschirm zum Beispiel mit PRINT CHR$(14) auf Kleinschrift umschalten. Das gleiche geht auch auf einem Commodore-Drucker. Senden Sie den Code CHR$(14) an den Drucker, so werden alle folgenden Zeichen in Groß-/Kleinschrift und nicht mehr in Graphic-/Großschrift ausgegeben. Das folgende kleine Beispiel veranschaulicht dies:

```
OPEN 4,4
PRINT#4,CHR$(14);"Das ist ein Test"
CLOSE 4
```

Es muß jedoch bei dieser Methode darauf geachtet werden, daß der Druckerbefehl in diesem Fall nur für den direkt nachfolgenden Text Gültigkeit hat. Beim Commodore 64 wird jeweils der gesamte Bildschirm umgeschaltet; bei einem Drucker gilt der Befehl nur für die nachfolgende Zeichenkette oder eine einzige Druckzeile. Wie Sie sehen, werden die einzelnen Befehle an den Drucker gesendet, wie jedes andere Zeichen auch. Der Drucker »weiß« dabei ganz genau, daß diese Zeichen nicht gedruckt werden können und führt den Befehl entsprechend aus. Es sei an dieser Stelle vielleicht darauf hingewiesen, daß bei Commodore und auch im amerikanischen ASCII-Zeichenstandard die Zeichen mit den Codes CHR$(0) bis CHR$(31) generell kein druckbares Zeichen, sondern Steuercodes darstellen, die bestimmte Funktionen übernehmen. Der Code 13 steht dabei beispielsweise für den Wagenrücklauf (carriage return), der auf dem Computer dem Drücken der RETURN-Taste entspricht. In Tabelle 1 sind die wichtigsten Steuercodes der Commodore-Drucker MPS 801, 802 und 803 zusammengefaßt.

Drucker-Steuercodes für Bedeutung	MPS 801/803	MPS 802
Bit-Muster-Modus ein (Drucken ohne Zwischenraum!)	CHR$ (8)	-
Zeilenvorschub (line feed)	(10)	(10)
Wagenrücklauf (carriage return) mit Zeilenvorschub	(13)	(13)
Sperrschrift/erweiterte Zeichen	(14)	(14)
Standardzeichen ein	(15)	(15)
Druckpositionsbestimmung	(16)	(16)
Textmodus ein (unshift)	(17)	(17)
Reverses Drucken ein (RVS on)	(18)	(18)
Seitenaufteilung aus	-	(19)
Wiederholen des Bitmusters	(26)	-
Druckstartposition – Punktadresse	(27)	-
Stringabschluß	-	(29)
Anführungszeichen	(34)	(34)
Wagenrücklauf ohne Zeilenvorschub	-	(141)
Grafik-Modus (hochgeschaltet: shift)	(145)	(145)
Reverses Drucken aus (RVS off)	(146)	(146)
Seitenaufteilung an	-	(147)
Seitenvorschub	-	(12)

Tabelle 1. Drucker-Steuercodes für MPS 801/803 und MPS 802

Es sei an dieser Stelle vielleicht darauf hingewiesen, daß der Wagenrücklauf (carriage return; CHR$(13)) und der Zeilenvorschub (line feed; CHR$(10)) normalerweise nichts miteinander zu tun haben. Bei Ihrem Computer sind beide Befehle miteinander verknüpft; und auch ein Commodore-Drucker führt beim Empfangen des Codes 13 einen Wagenrücklauf aus, wobei gleichzeitig in die nächste Zeile gesprungen wird. Im allgemeinen erfolgt bei einem Wagenrücklauf jedoch kein Zeilenvorschub, das heißt, der Cursor auf dem Bildschirm oder der Druckkopf beim Drucker würde auf den Anfang der gleichen Zeile gesetzt. Bei Druckern von Commodore ist das, wie gesagt, miteinander verknüpft. Für andere Druckertypen gilt das jedoch in der Regel nicht. Hier muß der Zeilenvorschub explizit angegeben oder durch Schalter im Inneren des Druckers so eingestellt werden, daß er bei einem Carriage Return automatisch erfolgt. Es sei in diesem Zusammenhang vielleicht auf eine Besonderheit der Commodore-Computer hingewiesen. Wenn Sie zum Beispiel ein Listing ausgeben wollen, so ist eine der gängigen Befehlszeilen dazu folgende:

```
OPEN 1,4: CMD 1: LIST <RETURN>
```

Hier eröffnen Sie eine Datei mit der Nummer 1 und sprechen darüber den Drucker mit der Geräteadresse 4 an. Der CMD-Befehl sorgt dabei dafür, daß das Listing, das üblicherweise auf dem Bildschirm erscheinen würde, nun auf dem (hoffentlich angeschlossenen) Drucker ausgegeben wird. Der Computer gibt dabei am Ende einer jeden Zeile ein Wagenrücklauf-Kommando an den Drucker, so daß an den Anfang der nächsten Zeile zurückgegangen wird. Eröffnen Sie jedoch eine Datei mit einer Nummer, die größer als 128 ist, so schickt der Computer zusätzlich zu dem normalen Wagenrücklauf noch ein Line-Feed-Kommando an den Drucker. Das erlaubt die Ausgabe eines Listings mit freigelassenen Zwischenräumen zwischen zwei Zeilen. Bei Druckern, die bei einem Carriage Return nicht automatisch auch ein Line Feed senden, erlaubt diese Anwendung einen einwandfreien Ausdruck, ohne daß die DIP-Schalter, die oft im Inneren des Druckers liegen, angetastet werden müssen. Zu dieser Befehlsfolge ein Beispiel:

```
OPEN 129,4: CMD 129: LIST <RETURN>
```

Außer den schon erwähnten Befehlscodes, die wie einfach zu druckende Zeichen an den Drucker übertragen werden,

gibt es bei einigen Commodore-Druckern und Kompatiblen, beziehungsweise Druckern mit einem Commodore-Interface noch eine andere Art der Befehlsübertragung; nämlich die Sekundäradressen im OPEN-Befehl.

Bei einem normalen Ausdruck wird die Sekundäradresse in der Regel nicht benötigt und deshalb auch nicht angegeben. Der Computer setzt dann automatisch eine Null. Diese Null ist jedoch auch für einen Commodore-Drucker ein Befehl und »sagt« diesem, daß er die nun folgenden Zeichen als Großbuchstaben und Grafikzeichen ausdrucken soll (es sei denn, der Befehl CHR$(14) fordert etwas anderes). In Tabelle 2 sehen Sie die Sekundäradressen der Commodore-Drucker mit ihren jeweiligen Funktionen aufgeführt.

Sekundäradressen im »Open«-Befehl beim MPS 802:

0	Drucken von Großbuchstaben und Grafikzeichen (Grafik-Modus)
1	Drucken der Daten nach einem vorher definierten Format
2	Speicherung der Formatierungsdaten
3	Angabe der Zeilenanzahl pro Seite
4	Absetzen der Format-Fehlerdiagnose-Nachricht
5	Definition eines programmierbaren Zeichens
6	Spezifizierung von Leerzeilen zwischen gedruckten Zeilen
7	Drucken von Klein- und Großbuchstaben (Textmodus)
8	nicht benutzt
9	Unterdrückung des Drucks von Fehlerdiagnose-Nachrichten
10	Drucker in Grundstellung zurücksetzen

Eine »Datei«-Nummer größer als 127 im »OPEN«-Befehl bewirkt im Ausdruck einen doppelten Zeilenvorschub; es wird also jeweils eine Leerzeile ausgegeben (zum Beispiel in einem Listing).

zum Beispiel OPEN 130,4,0

Tabelle 2. Sekundäradressen im »OPEN«-Befehl beim MPS 802

Wie schon erwähnt, wirkt der Code CHR$(14) nur jeweils für die direkt darauffolgenden Daten. Soll nun ein ganzer Text in Klein-/Großbuchstaben ausgedruckt werden, so ist die Verwendung der Sekundäradresse 7 sinnvoll. Hier wird der Drucker komplett in den Klein-/Großschriftmodus umgeschaltet, und es ist dann zum Beispiel möglich, ein Listing in Klein-/Großschrift auszudrucken:

```
OPEN 1,4,7: CMD 1: LIST <RETURN>
```

Die Bedienung der Drucker von Commodore, oder die Bedienung von Druckern mit einem Interface, das den entsprechenden Drucker einen Commodore-Drucker simulieren läßt, ist also recht einfach. Es werden sowohl Zeichencodes als auch Sekundäradressen für die Druckersteuerung verwendet. Anders verhält es sich bei Epson-Druckern und Epson-Kompatiblen. Hier erfolgt die Befehlsübergabe ein wenig anders, da diese Drucker in der Regel einen viel größeren Befehlsvorrat besitzen und die Anzahl der »normalen« Steuerzeichen begrenzt ist. Außerdem verfügen diese Drucker in der Regel über eine Centronics-kompatible Schnittstelle, um an die meisten Computer angeschlossen werden zu können. Diese Computer können aber in der Regel nicht mit Sekundäradressen arbeiten, so daß auch diese Möglichkeit der Befehlsübermittlung wegfällt.

Hier (im ESC/P-Standard) arbeitet man mit sogenannten Escape-Sequenzen. Das Escape kommt dabei vom ASCII-Steuerzeichen ESC, das auf den meisten Computern auch als eigene Taste vorhanden ist. Dabei wird jedoch nicht auf die schon bekannten Standardsteuerzeichen von 0 bis 31 verzichtet. Diese behalten ihre Funktion auch weiterhin, wobei jedoch nun der ASCII-Standard (ASCII = American Standard Code for Information Interchange) gilt, an den sich Commodore bei seinen Druckern und Computern nur sehr bedingt gehalten hat. Die einzelnen Steuerzeichen und ihre Bedeutung sind in Tabelle 3 dargestellt.

Das ESC-Zeichen stellt ein Steuerzeichen mit dem Code 27 ($1B) dar und kann vom C64 aus mit der Anweisung CHR$(27) an einen Drucker geschickt werden.

Soll nun ein Befehl an den Drucker gesendet werden, so wird das ESC-Zeichen übergeben. Danach folgt der eigentliche Befehlscode, der nun den gesamten Bereich der 256 möglichen Zeichen abdecken kann. Wie Sie sehen, ist also auf diese Art und Weise schon ein Druckerbefehlssatz von 256 Grundbefehlen möglich, wobei eventuelle weitere Befehlsunterteilungen nicht berücksichtigt sind. Wir wollen uns diese Art der Befehlsübergabe einmal etwas genauer betrachten. Sehen Sie sich dazu einmal die Tabelle 4 an. Sie enthält die wichtigsten Befehle der Epson- und kompatibler Drucker.

Als Beispiel schalten wir einmal in einen anderen Druckmodus; nämlich von Normalschrift auf Doppelanschlag. In diesem Modus werden von Epson-kompatiblen Druckern alle Zeichen zweimal übereinander gedruckt, wobei ein geringfügiger Zeilenvorschub erfolgt. Das Ergebnis ist eine stärkere und schönere Schrift:

```
OPEN 1,4
PRINT#1,CHR$(27);CHR$(71);
CLOSE 1
```

Der Code 71 entspricht außerdem einem großen »G«, weshalb der Befehl für den Doppelanschlag auch mit ESC G bezeichnet werden kann. Wollen wir den Doppelanschlag wieder rückgängig machen, so müssen wir nur den Code ESC H an den Drucker senden:

```
OPEN 1,4
PRINT#1,CHR$(27);CHR$(72);
CLOSE 1
```

Wie Sie sehen, ist auch im ESC/P-Standard die Befehlsübergabe an einen Drucker denkbar einfach. Es darf jedoch nicht vergessen werden, daß auch die normalen ASCII-Steuerbefehle noch vorhanden sind. Carriage Return beispielsweise hat nach wie vor den Code 13 und Line Feed entspricht der Codezahl 10. Ansonsten gelten die übrigen ASCII-Zeichen (Tabelle 3). Hat Ihr Drucker also beispielsweise einen eingebauten Summer, so können wir ihn dazu veranlassen, Töne von sich zu geben:

```
OPEN 1,4
PRINT#1,CHR$(7);
CLOSE 1
```

Wichtig für den Computeranwender ist aber natürlich noch eine weitere Eigenschaft von vielen Druckern: Der Grafik- oder Bitmapmodus.

Dieser Grafikmodus ist natürlich nur auf grafikfähigen Druckern möglich, was der Commodore MPS 802 zum Beispiel nicht ist! Außerdem muß beim Druck von Grafiken darauf geachtet werden, mit wievielen Nadeln der Matrixdrucker arbeitet. Beim C64 besteht nämlich jedes Zeichen aus 8 mal 8 Punkten; MPS 801 und 803 können jedoch nur maximal 7 Punkte in der Senkrechten ansteuern. Es fehlt hier also eine Punktreihe. Der Epson FX-80 wiederum verfügt über 9 Nadeln; er hat also eine »zuviel«.

Gerade im Grafikmodus der Drucker müssen wir zwischen Commodore-Druckern und Druckern von Fremdherstellern unterscheiden. Verwenden Sie den Drucker eines Fremdherstellers außerdem mit einem Interface für den seriellen Bus des Commodore 64, so sind außerdem die Eigenheiten des jeweiligen Interfacetyps zu beachten. Es sei an dieser Stelle auf unseren Artikel über Interfaces in der 64'er-Ausgabe 2 verwiesen, der die Eigenheiten der bekanntesten Interfaces vorstellt.

Im folgenden ein paar Beispiele, wie Sie auf mehreren Druckertypen eine Einzelnadelansteuerung vornehmen können.

Zuerst die beiden Commodore-Drucker MPS 801 und 803. Diese Drucker besitzen 7 Drucknadeln und können über den Code 8 ein freidefinierbares Sonderzeichen drucken. Bei der Definition dieses Sonderzeichens ist zu beachten, daß jeder

HEX		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	BIN	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	NUL 0	16	SP 32	0 48	§ 64 △	P 80	\ 96 △	p 112	NUL 128	144	SP 160	0 176	§ 192 △	P 208	\ 224	p 240
1	0001	1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113	129	DC1 145	! 161	1 177	A 193	Q 209	a 225	q 241
2	0010	2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114	130	DC2 146	" 162	2 178	B 194	R 210	b 226	r 242
3	0011	3	DC3 19	# 35 △	3 51	C 67	S 83	c 99	s 115	131	DC3 147	# 163 △	3 179	C 195	S 211	c 227	s 243
4	0100	4	DC4 20	$ 36 △	4 52	D 68	T 84	d 100	t 116	132	DC4 148	$ 164 △	4 180	D 196	T 212	d 228	t 244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	133	149	% 165	5 181	E 197	U 213	e 229	u 245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	134	150	& 166	6 182	F 198	V 214	f 230	v 246
7	0111	BEL 7	23	' 39	7 55	G 71	W 87	g 103	w 119	BEL 135	151	' 167	7 183	G 199	W 215	g 231	w 247
8	1000	BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120	BS 136	CAN 152	(168	8 184	H 200	X 216	h 232	x 248
9	1001	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121	HT 137	EM 153	) 169	9 185	I 201	Y 217	i 233	y 249
A	1010	LF 10	26	* 42	: 58	J 74	Z 90	j 106	z 122	LF 138	154	* 170	: 186	J 202	Z 218	j 234	z 250
B	1011	VT 11	ESC 27	+ 43	; 59	K 75	Ä 91 △	k 107	ä 123 △	VT 139	ESC 155	+ 171	; 187	K 203	Ä 219 △	k 235	ä 251 △
C	1100	FF 12	28	, 44	< 60	L 76	Ö 92 △	l 108	ö 124 △	FF 140	156	, 172	< 188	L 204	Ö 220 △	l 236	ö 252 △
D	1101	CR 13	29	- 45	= 61	M 77	Ü 93 △	m 109	ü 125 △	CR 141	157	- 173	= 189	M 205	Ü 221 △	m 237	ü 253 △
E	1110	SO 14	30	· 46	> 62	N 78	^ 94 △	n 110	ß 126 △	SO 142	158	· 174	> 190	N 206	^ 222 △	n 238	ß 254 △
F	1111	SI 15	31	/ 47	? 63	O 79	_ 95	o 111	DEL 127	SI 143	159	/ 175	? 191	O 207	_ 223	o 239	255

Tabelle 3. Die ASCII-Tabelle ist genormt (gilt aber nicht für Commodore-Drucker)

△ : Internationale Sonderzeichen anwählbar über <ESC R n> oder DIP-Schalter

Drucknadel ein Bit in einem Byte zugeordnet wird, wobei die oberste Nadel dem Bit 0 und die unterste Nadel dem Bit 6 eines Bytes entspricht. Da der MPS 801 und 803 mit einer 7 mal 6 Punktmatrix arbeitet, benötigen wir für die Definition eines Zeichens also 6 Byte. Bild 1 zeigt, wie so ein Zeichen durch 6 entsprechende Bytes definiert wird. Listing 1 zeigt, wie Sie das entsprechende Zeichen auch auf dem Drucker ausgeben können.

Bei Epson-kompatiblen Druckern funktioniert diese Ansteuerung ganz ähnlich; nur gibt es hier meistens verschiedene Grafikmodi, die von der Anzahl der Drucknadeln abhängen. Da viele Computer mit einer Zeichenhöhe von 8 Punkten arbeiten, haben Drucker im ESC/P-Standard die Möglichkeit, einen 8-Nadel-Grafikmodus zu realisieren. Der Drucker reagiert dann, als besäße er nur 8 Nadeln und kann so sehr einfach angesteuert werden.

Bei diesen Druckern macht sich der, in der Regel ziemlich große, Pufferspeicher positiv bemerkbar. Der Computer sendet die einzelnen Daten für das Bitmuster, wobei diese vom Drucker gespeichert werden. Erst wenn der Computer bis zu einer ganzen Druckzeile gesendet hat, werden die Werte umgesetzt und ausgedruckt.

Im ESC/P-Standard schaltet das Kommando ESC * den Bitmustermodus für 8 Nadeln ein. Direkt hinter diesem Befehl folgt die Einstellung des Druckmodus (siehe Handbuch des Druckers) und danach die Anzahl der Bitmuster/Bytes, die gesendet werden sollen. In Listing 2 sehen Sie ein kleines Beispielprogramm, das auf einem Epson-Drucker ein Bitmuster erzeugt.

Diese ganzen Angaben über spezielle Befehle sind natürlich mit Vorsicht zu genießen, wenn Drucker anderer Hersteller oder irgendwelche »exotischen« Interfaces verwendet werden. Es sollte hier nur ein kleiner Einblick in die Anwendung von Steuerzeichen auf Druckern gewährt werden, der zum Experimentieren anregen soll. Es ist wohl eine der schönsten Einsatzmöglichkeiten eines Druckers, wunderbare Grafikbilder oder seltsame Schriftarten zu drucken, wobei der Kreativität des Benutzers kaum Grenzen gesetzt sind.

Natürlich konnten wir Ihnen hier keine komplette Drucker-Schulung geben. Trotzdem hoffen wir, in Ihnen die »Lust« aufs Programmieren eines Druckers geweckt zu haben. Für den eingefleischten Drucker-Liebhaber gibt es ohnehin nichts schöneres, als den Drucker zu immer neuen Kunststücken zu bewegen.

(Karsten Schramm/aw)

```
0 * * * 0 0
* 0 0 0 * 0
* 0 * * 0 0
* 0 0 0 * 0
* * * * 0 0
* 0 0 0 0 0
* 0 0 0 0 0

Bit 0 ( 1)   0     1     1     1     0     0
Bit 1 ( 2)   1     0     0     0     1     0
Bit 2 ( 4)   1     0     1     1     0     0
Bit 3 ( 8)   1     0     0     0     1     0
Bit 4 (16)   1     1     1     1     0     0
Bit 5 (32)   1     0     0     0     0     0
Bit 6 (64)   1     0     0     0     0     0
           =127 = 17 = 21 = 21 = 10 =  0
           +128 +128 +128 +128 +128 +128
            255  145  149  149  138  128
```

Bild 1. So definiert man ein »ß« für den MPS 801/803

```
10 DATA 255,145,149,149,138,128
20 FOR I = 1 TO 6
30  READ A: A$=A$+CHR$(A)
40 NEXT I
50 OPEN 1,4
60 PRINT#1,CHR$(8);A$;CHR$(15);"= SCHARFES S"
70 CLOSE 1
```

Listing 1. Das »ß« wird gedruckt

```
10 OPEN 4,4
20 PRINT#4,CHR$(27);"*";CHR$(0);CHR$(100)CHR$(0);
30 FORX=129TO229
40 PRINT#4,CHR$(X);
50 NEXT
60 CLOSE4
READY.
```

Listing 2. Grafikdefinition nach dem ESC/P-Standard.



ESC/P-Steuercodes

Funktion	Code
Drucken und Zeilenvorschub	
– Druckkopfrücklauf	CR
– Zeilenvorschub	LF
– Zeilenvorschub um n/180″ (24 Pin)	ESC J
– Zeilenvorschub um n/216″ (9 Pin)	ESC j
Formatsteuerung	
I. Vertikal	
(1) Zeilenabstand	
– Zeilenabstand 1/8″	ESC 0
– Zeilenabstand 1/6″	ESC 2
– Zeilenabstand n/180″ (24 Pin)	ESC 3
– Zeilenabstand n/60″ (24 Pin)	ESC A
– Zeilenabstand 7/72″ (8 Pin)	ESC 1
– Zeilenabstand n/216″ (8 Pin)	ESC 3
– Zeilenabstand n/72″ (8 Pin)	ESC A
(2) Formularvorschub und Formularlänge	
– Formularvorschub	FF
– Formularlänge in Zeilen	ESC C
– Formularlänge in Zoll	ESC C 0
(3) Vorschub auf neues Formular	
– Vorschub auf neues Formular setzen	ESC N
– Vorschub auf neues Formular löschen	ESC 0
(4) Vertikale Tabulierung	
– Vertikaler Tabulator	VT
– Vertikale Tabulatorsprünge setzen	ESC B
– Auswahl eines VFU-Kanals	ESC /
– VFU-Position setzen	ESC b
II. Horizontal	
(1) Rand	
– Rechten Rand setzen	ESC Q
– Linken Rand setzen	ESC 1
(2) Horizontale Tabulierung	
– Horizontaler Tabulator	HT
– Horizontalen Tabulator setzen	ESC D
(3) Zeichenzwischenraum	
– Zeichenzwischenraum setzen	ESC (space)
(4) Punktposition	
– Absolute Punktposition setzen	ESC $
– Relative Punktposition setzen	ESC \
(5) Randausgleich	
– Automatischer Randausgleich	ESC a
III. Horizontal und Vertikal	
– Horizontale/Vertikale Tab.Einheiten	ESC e
– Horizontale/Vertikale Tab.Sprünge	ESC f
Drucksteuerung	
I. Druckqualität	
– Umschalten zwischen Entwurfs-/Schönschriftqualität	ESC x
II. Schriftart	
– Umschalten auf Schriftart »Elite«	ESC M
– Schriftart »Elite« löschen bzw. »Pica« setzen	ESC P
– Proportionalschrift-Modus setzen/löschen	ESC p
– Wahl der Schriftartenfamilie	ESC k
III. Druckmodi	
(1) Breitschrift	
– Breitschrift-Modus mit autom. Rückschaltung	SO
– Breitschrift-Modus mit autom. Rückschaltung	ESC SO
– Breitschrift-Modus mit autom. Rückschaltung löschen	DC 4
– Breitschrift-Modus setzen/löschen	ESC W
(2) Engschrift	
– Engschrift-Modus setzen	SI
– Engschrift-Modus setzen	ESC SI
– Engschrift-Modus löschen	DC 2
(3) Fettschrift	
– Fettschrift-Modus setzen	ESC E
– Fettschrift-Modus löschen	ESC F
(4) Doppelanschlag	
– Doppeldruck-Modus setzen	ESC G
– Doppeldruck-Modus löschen	ESC H
(5) Kursivschrift	
– Kursivschrift-Modus setzen	ESC 4
– Kursivschrift-Modus löschen	ESC 5
(6) Unterstreichen	
– Unterstreichungs-Modus setzen/löschen	ESC –
(7) Potenzierung/Indizierung	
– Potenzierungs-Modus setzen	ESC S 0
– Indizierungs-Modus setzen	ESC S 1
– Potenzierungs-/Indizierungs-Modus setzen/löschen	ESC T
(8) Auswahlmodus	
– Druckmodi auswählen	ESC !
Zeichensätze	
– Anwählen internationaler Zeichensätze	ESC R
– Sonderzeichen ASCII-Code 128-159D	ESC 6
– Steuercodes ASCII-Code 128-159D	ESC 7
– Internationale Zeichentabelle	ESC I
– Spezialzeichensatz	ESC m
– Wahl des ESC/P-Tabelle (Kursiv/Grafik)	ESC t
Bitmuster	
– Bitmuster-Modus in Normaldichte setzen	ESC K
– Bitmuster-Modus mit doppelter Dichte setzen	ESC L
– Bitmuster-Modus mit doppelter Dichte und doppelter Geschwindigkeit setzen	ESC Y
– Bitmuster-Modus mit vierfacher Dichte setzen	ESC Z
– Bitmuster-Modus anwählen	ESC *
– Bitmuster-Modus zuordnen	ESC ?
– 9-Punkt-Bitmuster-Modus	ESC Y
Ladbare (benutzerdefinierbare) Zeichen	
– Ladbaren Zeichensatz anwählen/abwählen	ESC %
– Ladbare Zeichen definieren	ESC &
– ROM-Zeichensatz kopieren	ESC : 0
Verschiedene Codes	
– Summer	BEL
– Rückschritt	BS
– Drucker initialisieren	ESC @
– Papierende-Erkennung abschalten	ESC 8
– Papierende-Erkennung einschalten	ESC 9
– Druckkopf in Ausgangsstellung bringen	ESC <
– Druckrichtung wählen	ESC U
– Drucken mit halber Geschwindigkeit setzen/löschen	ESC s
– Kontrolle des Ausdrucks	ESC i
– Autom. Einzelblatteinzug ein/aus	ESC EM
Eingabedaten-Steuerung	
– Löschen des Druckpuffers	CAN
– Löschen des letzten druckbaren Zeichens	DEL
– Drucker anwählen	DC 1
– Drucker abwählen	DC 3
– Modus-Druckdaten wiederholen/setzen	ESC V
– MSB-Steuerung löschen	ESC #
– MSB-Steuerung setzen	ESC >
– MSB-Steuerung zurücksetzen	ESC =
Ladbarer Zeichengenerator	
– Zeichengenerator definieren	ESC&
– Ein-/Ausschalten ladbaren Zeichengenerator	ESC%
– Kopieren internen ZG auf ladbaren ZG	ESC:0

Tabelle 4. Die wichtigsten ESC/P-Steuercodes (ESC/P = Epson-Standard-Code for Printers)

64'er HARDWARE-SERVICE

Bestellungen aus anderen Ländern bitte per Auslandspostanweisung!

Bestellungen aus der Schweiz bitte direkt an:
Markt&Technik Vertriebs AG
Kollerstrasse 3
CH-6300 Zug
Tel. 042/41 56 56

Bestellungen aus Österreich bitte direkt an:
Ueberreuter Media
Handels- und Verlagsges. mbH,
Alser Straße 24,
1091 Wien
Tel. 0222/48 15 38-0

Hardware für alle – ein neuer 64'er Leser-Service

Der Commodore 64 hat schon oft bewiesen, wie vielseitig er ist. Er läßt sich nicht nur mit Programmen, sondern auch durch so manche Hardware-Erweiterung sinnvoll nutzen und ausbauen. Dabei ist es sicherlich ein reizvoller Bestandteil des Computer-Hobbys, sich solche Erweiterungen selbst nachzubauen. Aber nicht jeder Leser verfügt über die Gelegenheit und Zeit zur Platinenherstellung. Hinzu kommt, daß es oft zu teuer ist, wegen einer bestimmten Erweiterung Investitionen von mehreren hundert Mark für eine Platinenstation zu tätigen. Die in der 64'er abgedruckten Hardware-Erweiterungen sind in drei verschiedenen Ausbaustufen zu erhalten:

1. Als Platinen
Nur Leerplatinen. Die Beschaffung der Bauteile und der Zusammenbau bleiben bei Ihnen.

2. Als Bausätze
Unsere Bausätze enthalten alle Teile, die notwendig sind, um die beschriebene Erweiterung komplett aufzubauen. Sie brauchen die Bauteile nur noch, gemäß der Anleitung im Heft, zusammenzulöten und einzubauen.

3. Als Fertiggeräte
Die Fertiggeräte sind komplett aufgebaute und geprüfte Geräte. Sie brauchen die Erweiterung lediglich noch einzubauen.

Qualität & Service

- Die 64'er Hardware hat einen hohen Qualitätsstandard. Wir verwenden nur beste Epoxid-Harz-Platinen mit Lötstopp-Lack.
- Wir verwenden nur Präzisionssockel mit gedrehten Kontakten.
- Alle Platinen werden professionell gefertigt. Wenn notwendig mit doppelseitiger Beschichtung und Löt-Durchkontaktierungen.
- Jedes Gerät, das wir versenden, wurde auf Funktionstüchtigkeit geprüft.
- Wir sind auch nach dem Verkauf für Sie da. Neben der gesetzlichen Garantie bietet unser Service- und Fertigungspartner Ihnen Hilfe und Unterstützung an.

Einbauservice

Für die Angebote 4 (Super Kernal) und 5 (64'er DOS) bieten wir einen Einbauservice an. Jeder Lieferung dieser Produkte liegt neben der detaillierten Einbauanleitung ein Angebot zum kostengünstigen Umbau Ihres C64 beziehungsweise Ihrer 1541 Floppy bei. Falls Ihr C64 keine gesockelten Bausteine besitzt, können Sie dort ebenfalls hochwertige Stecksockel einbauen lassen.

Unsere Garantie

Im Rahmen der Versand- und Lieferbedingungen unterliegen die Geräte einer Gewährleistungszeit von 6 Monaten ab Lieferung. Der Lieferung liegt eine Service-Karte bei, die Sie im Falle einer Beanstandung zusammen mit dem Gerät an die auf der Karte vermerkte Adresse schicken können. Die gleiche Karte verwenden Sie bitte bei Reparaturen nach der Garantiezeit.

Unser Angebot

Angebot 1: Expansion-Port EPROM-Platine mit 1 x 8 KByte Speicherplatz für 2732 bis 2764 EPROMS.

Beschreibung in Ausgabe 10/85
Bestellnummer: HW 010
DM 19,80* (sFr. 17,50)
Dieser Artikel wird nur als Fertiggerät angeboten.

Angebot 4: Super Kernal

Erweitertes Betriebssystem für den C64 mit vielen neuen Funktionen, inkl. Adaptersockel, einbaufertig in den C64.
Beschreibung in Ausgabe 11/85

Version 1: Enthält Hypra Load / DOS 5.1 / Funktionstastenbelegung / Renew / RS232
Bestellnummer: HW 040

Version 2: Enthält Hypra Load / DOS 5.1 / Funktionstastenbelegung / Renew / Super Centronics Schnittstelle
Bestellnummer: HW 041

Version 3: Enthält Hypra Load / DOS 5.1 / Funktionstastenbelegung / Renew / Hypra Save
Bestellnummer: HW 042

Version 4: Enthält Hypra Load / DOS 5.1 / Funktionstasten / Hypra Save / Centronics klein
Bestellnummer: HW 043

Jede Version kostet:
DM 39,80* (sFr. 34,-)

Angebot 5: 64'er DOS

Jetzt wird das 1541 Laufwerk zum Renner. Mit wenig Aufwand beschleunigt 64'er DOS alle Funktionen des Laufwerkes. Das neue Betriebssystem für den Commodore 64 und das 1541 Laufwerk ist auf 2 Speicher-EPROMs der Sorte 2764 untergebracht und inkl. Adaptersockel einbaufertig vorbereitet.
Beschreibung in Ausgabe 3/86 (Einbauanleitung liegt bei).
Preis für beide EPROMs inkl. Adaptersockel
Bestellnummer: HW 050
DM 69,-* (sFr. 59,-)
Lieferbar ab April 1986

C64'er-Fastkernal inkl. Adaptersockel
Bestellnummer: HW 051
DM 39,80* (sFr. 34,-)
Lieferbar ab April 1986

1541-Fast DOS inkl. Adaptersockel
Bestellnummer: HW 052
DM 39,80* (sFr. 34,-)
Lieferbar ab April 1986

Angebot 7: Akustikkoppler

Der HITRANS 300 C stach im Akustikkoppler-Test der Ausgabe 3/86 durch die besten Übertragungseigenschaften hervor. Sie erhalten ihn bei uns als Fertiggerät, lediglich eine Blockbatterie muß eingesetzt und das Gehäuse zugeschraubt werden. Sie können den Koppler auch über ein 12-Volt-Netzteil, das in jedem Elektronikgeschäft preisgünstig erhältlich ist, betreiben. Die Bauanleitung für ein RS232-Interface finden Sie in der Ausgabe 3/85.

Preis für Akustikkoppler HITRANS 300 C (ohne Batterie)
Bestellnummer: HW 071
DM 248,-* (sFr. 225,-)

Betriebssoftware auf Diskette
Bestellnummer: HW 071
DM 14,80* (sFr. 13,90)
Die Betriebssoftware befindet sich auch auf der Programm-Service-Diskette des 64er-Sonderheftes SH7/85.

Angebot 8: C-MOS-Hardware

Die C-MOS-RAM-Platine ist eine hervorragende Hilfe für Software-Entwickler. Da sich die verschiedensten EPROMs, PROMs und ROMs direkt simulieren lassen, können Sie beispielsweise neue Betriebssysteme und Zeichensätze entwerfen, ohne immer wieder EPROMs zum Ausprobieren brennen zu müssen. Sie können aber auch ein Programm wie etwa einen Monitor im CMOS-RAM ablegen und es mit einem Schalter in den Speicher Ihres Computers einblenden. Eine genaue Funktions- und Schaltungsbeschreibung finden Sie im 64'er, Ausgabe 4/86.

Leerplatine
Bestellnummer: HW 080
DM 49,80* (sFr. 43,-)

Kabel inkl. Stecker
Bestellnummer: HW 081
DM 24,80* (sFr. 22,-)

Bausatz mit Akku und Jumper (inkl. Kabel mit Stecker)
Bestellnummer: HW 082
DM 159,80* (sFr. 139,-)

Fertiggerät in Luxusausführung. Mikroschalter anstelle der Jumper erhöhen die Bedienerfreundlichkeit, und eine Siliziumbatterie sichert die Daten über mindestens 5 Jahre.
Bestellnummer: HW 083
DM 198,80* (sFr. 169,-)

* inkl. MwSt. Unverbindliche Preisempfehlung

Bitte verwenden Sie für Ihre Bestellung immer die abgedruckte Postgiro-Zahlkarte oder einen Verrechnungsscheck. Sie erleichtern uns damit die Auftragsabwicklung, und dafür berechnen wir Ihnen keine Versandkosten.

Das Computerbüro

Heimcomputer lassen sich als professionelle Geräte im Büro einsetzen – vorausgesetzt man hat sich die richtige Software zugelegt. Einige »professionelle« Programme aus den Bereichen Textverarbeitung, Datenverwaltung und Tabellenkalkulation sollen vorgestellt werden.

Wird einem Computeranfänger im Heim- oder Kleinbetriebsbereich ein Profiprogramm, wie Textverarbeitung oder Kalkulation angeboten, so stößt der Anbieter meistens auf Ablehnung. Diese Haltung des Anwenders rührt meistens aus der Unkenntnis der Anwendungsmöglichkeiten dieser Programme her. Dabei ist es gerade ein solches Programm, das einem Computer zu ungeahnten Leistungen im Verwaltungsbereich verhilft. Ist ein Anwender zu dieser Erkenntnis gekommen, so macht er leicht den naheliegenden Fehler, zuerst den Computer und dann das Programm zu kaufen. In einigen Fällen geht das gut aus, dennoch wird es meist so sein, daß das Programm nicht den gestellten Anforderungen entspricht. Ein Anwender, der lediglich ab und zu Briefe schreibt, ist mit einem Textprogramm der mittleren Klasse gut bedient. Er wird nichts von einem Programm mit allen nur denkbaren Raffinessen haben, denn solche Programme erfordern einen teuren Hardwareaufwand und sind oft nicht einfach zu bedienen.

Andererseits wird ein Schriftsteller, der jeden Tag mit dem Textsystem arbeitet, die Bedienungsfunktionen schnell im Griff haben. Da er seine Texte oft ändern muß, kommen ihm die mannigfaltigen Möglichkeiten sehr zugute.

Um die Kaufentscheidung für das richtige Programm zu vereinfachen, haben wir verschiedene Programme für den Commodore 64 und Commodore 128 PC zusammengestellt und ihre Haupteinsatzgebiete herausgearbeitet. Die hier beschriebenen Programme sind in drei Gruppen einzuteilen: in Textverarbeitungsprogramme, Datenbanken und Kalkulationsprogramme.

Da die Textverarbeitung oft gebraucht wird, findet man hier die größte Anzahl von Programmen. Getestet wurden Vizawrite 64, Textomat Plus und Star Texter für den Commodore 64 sowie Wordstar, Superscript und Protext-128 für den Commodore 128.

Vizawrite für den C64

Das wohl beste, aber auch teuerste Textverarbeitungsprogramm für den Commodore 64 ist Vizawrite. Mit diesem Programm ist auch mit der auf 40 Zeichen Breite eingeschränkten Bildschirmdarstellung fast professionelle Textverarbeitung möglich. Der Mangel wird dadurch kompensiert, daß der Bildschirm wie ein Fenster über den Text verschoben wird. Auf diese Weise ist es möglich, bis zu 240 Zeichen pro Zeile zu editieren. Das hat den Vorteil, daß der Text bei der Erstellung gleich so formatiert werden kann, wie er später ausgedruckt werden soll. Dazu hat der Anwender außer einem flexiblen Editor Tabulatoren, Dezimaltabulatoren, Zentrierung, Einrückbefehle und Blocksatz zur Verfügung (Blocksatz ist das Ausrichten des rechten Textrandes).

Eine weitere Hilfe bietet das sogenannte Word-wrapping. Dabei wird ein Wort, das nicht mehr in die gerade bearbeitete Zeile paßt, komplett in die nächste übernommen. Das verhindert die Entstehung von schwer les- und editierbaren Wortleichen am Ende einer Zeile. Der zu erstellende Text darf bei Vizawrite zirka 32000 Zeichen umfassen, was etwa acht bis neun vollgeschriebenen Schreibmaschinenseiten entspricht. Das ist weit mehr, als die meisten Programme bieten.

Ist ein Text fertig erstellt, so empfiehlt es sich, ihn zur Korrektur im Zusammenhang zu lesen. Da sich das mit dauerndem Bildschirmverschieben äußerst mühselig gestalten würde, gibt es die Möglichkeit, mit einem Befehl den gesamten Text auf 38 Zeichen Breite umzuformatieren. Die Editiermöglichkeiten bleiben dabei erhalten. Nachdem alle Fehler beseitigt sind, wird die Umformatierung wieder rückgängig gemacht.

Der Ausdruck von Texten gestaltet sich bei Vizawrite sehr flexibel, denn es werden eine Reihe von Druckern und Schreibmaschinen standardmäßig unterstützt. Sollte man dennoch im Besitz eines Ausgabegerätes sein, das nicht im Druckermenü enthalten ist, so kann man auf die Definition von Druckersteuerzeichen zurückgreifen. In der sogenannten Formatzeile (die beliebig oft verändert werden kann) können jeweils bis zu zehn Steuersequenzen vereinbart werden, um etwa den Ausdruck auf Papier ganz den individuellen Anforderungen anpassen zu können (zum Beispiel Sperrschrift).

Ist der Druckmodus aufgerufen, können die meisten Einstellungen zum Ausdruck verändert werden, wie Randausgleich, Papierlänge, Druckertyp, etc. Zusätzlich kann der Anwender Kopf- und Fußzeilen definieren. Dies ermöglicht zum Beispiel automatische Seitennumerierung, immer wiederkehrende Erklärungen, Tabellenüberschriften, etc. Hier offenbart sich noch ein Vorteil des Systems: Diese Einstellungen müssen nicht jedesmal neu vorgenommen werden, sondern sie werden mit dem Schriftstück gespeichert, ebenso Farbeinstellungen und Formatierungsbefehle. Dies vereinfacht die Weiterverarbeitung von schon erstellten Texten erheblich. Der Ausdruck von Vizawrite muß nicht über die serielle Schnittstelle des Commodore 64 erfolgen, auch Drucker mit Centronics-Anschluß (Parallelübertragung) werden angesprochen. Dazu benötigt man ein Kabel, dessen eines Ende in den User-Port gesteckt wird, das andere in die Centronics-Buchse des Druckers. Dieses Kabel ist für zirka 50 Mark in allen Fachgeschäften erhältlich.

Das gleiche Kabel braucht man, um diese Drucker bei den Programmen für den Commodore 128 betreiben zu können. An Befehlen stellt Vizawrite zur Verfügung:

Verschieben und Kopieren von Textblöcken, Laden von Textbausteinen von Diskette, Suchen und Ersetzen, Löschen und Einfügen von Worten oder ganzen Zeilen und Diskettenbedienung. Sämtliche Befehle werden durch das Drücken zweier Tasten aufgerufen. Dadurch wird die Bedienung sehr schnell und einfach. Zum Laden von Textbausteinen ist noch hinzuzufügen, daß diese nicht unbedingt mit Vizawrite erstellt sein müssen. Es werden viele Fremdformate unterstützt, was auch die Weiterverarbeitung von Texten anderer Programme (Assembler, Multiplan, etc.) ermöglicht.

Eine für den Geschäftsbereich besonders wichtige Eigenschaft ist die Möglichkeit der Erstellung von Serienbriefen. Dabei werden die benötigten Daten wahlweise aus einem dafür reservierten Speicher oder von Diskette geholt.

Insgesamt betrachtet, ist Vizawrite 64 das ideale Programm für den Vielschreiber, der nicht zuviel Geld in die Hardware investieren will oder kann. Das Geld, das der Computer weniger kostet, kann für den Drucker ausgegeben werden,

an dem sowieso nicht gespart werden sollte, denn er hat einen ganz entscheidenden Einfluß auf das Aussehen des fertigen Schriftstückes.

Textomat für den C64

Das Textverarbeitungsprogramm Textomat Plus besitzt Eigenschaften, die man bei keinem anderen Programm findet. Gleich beim Programmstart offenbaren sich dem Anwender die vielfältigen Einstellmöglichkeiten. Bevor ein Wort geschrieben wird, sind der Bildschirmzeichensatz und die Druckeranpassung zu laden. Da bei Textomat die Möglichkeit vorhanden ist, den Bildschirm und Druckerzeichensatz zu verändern (das hat natürlich nur bei Matrixdruckern einen Sinn), hat der Anwender keine Schwierigkeiten, Zeichen zu Papier zu bringen, die nicht dem Standard entsprechen. Die Zeichen werden mit einem einfach zu bedienenden Hilfsprogramm geändert. Jedes Zeichen ist aus Punkten zusammengesetzt, und man kann mit dem Zeicheneditor die einzelnen Punkte setzen oder löschen. Dies ist eine große Hilfe für alle Anwender, die Sonderzeichen häufiger benötigen, wie beispielsweise Techniker. Soll eine etwas größere Grafik in den Text eingefügt werden, so braucht man sich beim Textomat Plus nicht mit einzelnen Druckzeichen abzugeben, sondern lädt einfach die Grafik in den Textspeicher, wobei allerdings bestimmte Regeln beachtet werden müssen, die zumindest Anfängern Probleme bereiten. So ist es nicht einfach möglich, mit einem Malprogramm erstellte Grafiken zu laden. Trotzdem war Textomat Plus das erste Programm, mit dem Text und Grafik verarbeitet werden konnten. Leider sind diese Abfragen am Anfang für den Neuling etwas verwirrend. Nach dem Programmstart mit »LOAD"*",8,1« werden Fragen nach dem Druckersteuersatz und dem Bildschirmzeichensatz gestellt. Für unerfahrene Anwender empfiehlt es sich daher, bei jeder Frage einfach »F1« zu drücken (Bestätigung), bis man in den Schreibmodus kommt.

Für jemanden, der zum ersten Mal ein Textprogramm benutzt, sind vorrangig die Möglichkeiten zur Textmanipulation wichtig. Bei Textomat Plus sind diese Funktionen nicht gerade benutzerfreundlich geraten. Es gibt zwar ähnliche Funktionen wie bei Vizawrite, aber ihre Anwendung ist aufwendiger. Während bei Vizawrite einfach die Textteile, die kopiert, gelöscht oder verschoben werden sollen, mit dem Cursor markiert wurden, ist die Handhabung des Textomats komplizierter. Diese Befehle funktionieren nur zeilenweise, man muß also immer nach der Ausführung dieser Funktionen den überschüssigen Text löschen. Das erleichtert die Arbeit nicht gerade, das Aufrufen der Funktionen ist aber für den unerfahrenen Computerbenutzer gut gelöst. Alle Befehle lassen sich über Menü auswählen. Da diese Menüs immer aussagen, was der gerade aufgerufene Befehl bewirkt, kommt man schnell ohne Handbuch aus. Das hat Vorteile, wenn Sie nur ein Gelegenheitsschreiber sind, denn das Programm läßt sich so ohne Probleme bedienen. Für die, die öfter oder gar professionell mit dem Computer schreiben wollen, eignet sich Textomat Plus nicht so sehr, denn die Menütechnik kostet viel Zeit. Zusätzlich behindert die zeilenweise Ausführungsweise der Befehle ein schnelles Schreiben. Der Textomat Plus ist besonders für den Anwenderkreis geeignet, der gerne hochauflösende Grafik und Text mischen oder damit experimentieren möchte, oder auch Datenfernübertragung (DFÜ) durch das Telefon mit diesem Programm bewältigen will. Der integrierte Terminalmodus erlaubt es, beispielsweise in Mailboxen (sozusagen elektronische Briefkästen) herumzustöbern.

Star Texter, die Textverarbeitung aus dem Sybex-Verlag, wird komplett mit sehr gutem deutschen Handbuch, Diskette und einem Funktionsangebot geliefert, das zu dem Preis von 64 Mark seinesgleichen sucht. Da in Deutschland entwickelt, sind klare deutsche Kommandos und eine an die DIN-Norm angelehnte Tastenbelegung mit den deutschen Sonderzeichen (ö,ä,ü,ß) selbstverständlich. Es ist äußerst angenehm, daß man sich sofort nach dem Laden im Schreibmodus befindet, ohne sich erst durch ungezählte Menüs zu kämpfen. Alle notwendigen Funktionen und Einstellungen werden während der Arbeit mit Star Texter aufgerufen, beziehungsweise vorgenommen.

Immer gleichbleibende Parameter, wie der Druckertyp, der Zeichensatz, Tastaturbelegungen, Diskettenlaufwerke und Farbeinstellungen werden, nachdem sie einmal festgelegt wurden, automatisch mitgeladen.

Star Texter für den C64

Wer schon mit anderen Textprogrammen gearbeitet hat, weiß, daß es Funktionen gibt, die häufig benötigt und andere, die zwar wichtig sind, aber eben seltener gebraucht werden.

Star Texter verfügt über alle Grundfunktionen. Editieren, Verschieben, Löschen und Kopieren funktionieren schnell und korrekt. Leider sind diese Operationen teilweise blockorientiert, das heißt, sie sind immer nur auf ganze Zeilen anwendbar. Während es durchaus möglich ist, beliebig viele Worte mitten in einen Text einzufügen, können Textteile immer nur als ganze Zeilenblöcke verschoben werden.

Neben der Qualität der Grundfunktionen gewinnen die Sonderfunktionen, die für manche Anwendung besonders wichtig sind, an Bedeutung. Gerade in diesem Punkt hebt sich Star Texter von seiner Konkurrenz ab. Zunächst fällt auf, daß es zu keinem Zeitpunkt notwendig ist, die Systemdiskette nochmals einzulegen.

Wie fast alle bekannten Textprogramme versucht auch Star Texter, die 40-Zeichen-Darstellung zu umgehen. Das Schreibfeld besteht aus 21 Zeilen und bis zu 80 Spalten. Der Bildschirm wird dabei horizontal verschoben. Ein kleines Fenster am oberen Bildschirmrand gibt jederzeit Aufschluß über die aktuelle Cursorposition in einer Zeile (Bild 1). Andere Fenster informieren über die aktuelle Zeile und Spalte sowie über den jeweiligen Befehlsmodus. Da der Text nach der 21. Zeile ebenfalls vertikal verschoben wird, bis die maximale Zeilennummer 250 erreicht ist, kann man auch von einem großen Arbeitsblatt sprechen. Dieses Arbeitsblatt besteht aus 20000 Zeichen oder umgerechnet etwa sechs bis sieben Schreibmaschinenseiten. Wer mit diesem Textspeicher nicht auskommt, kann einzelne Schriftstücke aneinanderbinden.



Bild 1. Sonderzeichen und neue Zeichensätze sind für den Star Texter kein Problem

Erfreulich ist in diesem Zusammenhang, daß Star Texter auch bei fast gefülltem Textspeicher kaum langsamer wird.

Manchmal ist es von Nutzen, ein Schriftstück so betrachten zu können, wie es später auf dem Papier ausgedruckt wird. Nur so läßt sich feststellen, ob die Absätze stimmen, die Überschriften korrekt sind und der gesamte Text harmonisch verteilt ist. Star Texter bietet dafür einen sehr sinnvollen, weil einfach zu bedienenden, 80-Zeichen-Modus an. Durch einfaches Drücken auf die CBM- und SHIFT-Taste wird der gesamte Text umformatiert und kann in seiner Originaldruckbreite gelesen werden. Durcksonderfunktionen, wie beispielsweise Fettschrift und Unterstreichungen, bleiben (auf dem Bildschirm) allerdings unberücksichtigt. Der 80-Zeichen-Modus ist vor allem dann von großem Nutzen, wenn es um die Erstellung von Tabellen und formatierten Texten geht. Für diese Aufgabe bietet Star Texter noch weitere sehr sinnvolle Ergänzungen an: die Rechenfunktionen. Es sind alle Rechenoperationen und die im Commodore-Basic üblichen Operationen und Befehle gestattet. Der eigentliche Clou an diesem Modus ist seine Programmierbarkeit. Mit einem Befehlsspeicher von 3325 Byte lassen sich schon manche nützliche Routinen programmieren und im Text einbinden. So werden Anwendungen denkbar, bei denen im Text verschiedene Werte errechnet und anschließend weiterverarbeitet werden. Das Schreiben von Rechnungen, Listen und Karteikarten könnten solche Anwendungen sein.

Bei der Frage, wie ein bedienungsfreundliches Programm aussehen soll, gehen die Meinungen stark auseinander. Man könnte das hier angewendete Prinzip auch als Seitenkonzept bezeichnen, denn Star Texter verwendet nur drei Hauptmenüs und ein kleines Untermenü. Alle Hauptmenüs werden mit den Funktionstasten aufgerufen und sind nach Themenkreisen geordnet. Es gibt ein Disketten-, ein Drucker- und ein Sammelmenü, das selbst wiederum aus drei Seiten besteht. Während das Floppy- und Druckermenü ihre Funktion selbst erklären, sind im Sammelmenü alle nicht direkt zugeordneten Funktionen vereinigt. Einige dieser insgesamt 30 Einstellparameter werden von kaum einem bisher bekannten Programm angeboten. Beispielsweise der Unterpunkt Zeichensatz: Auf der Programmdiskette werden drei vorprogrammierte Zeichensätze (CBM, Atari, Futura) mitgeliefert. Diese Zeichensätze kann man als Beispiele ansehen, denn prinzipiell ist Star Texter in der Lage, jedes beliebige Zeichen abzubilden. Damit der Entwurf eines neuen Zeichensatzes nicht zum Problem wird, hat man auf der Diskette gleich auch ein Hilfsprogramm abgelegt. Es nennt sich Star Font, ist sehr komfortabel und läßt fast jede Manipulation des Zeichensatzes zu.

Die Palette der Anwendungen dieses Modus reicht von einzelnen wissenschaftlichen Zeichen bis hin zum kompletten Zeichensatz. Das ganze Entwerfen eines Zeichensatzes könnte natürlich als netter Gag bezeichnet werden, wenn es da nicht noch eine andere Funktion gäbe, mit der die neuen Zeichen auch auf dem Drucker verewigt werden könnten. Star Texter verfügt über einen Grafikdruck, der zwar deutlich langsamer aber dafür eben jedes beliebige Zeichen drucken kann. Allerdings wird der Wert der Funktion dadurch geschmälert, daß selbst auf einem Epson- oder Star-Drucker alle Zeichen (auch die nicht geänderten) wackelig ausgegeben werden, wie es von der MPS-802-Grafik her bekannt ist.

Formatieren leicht gemacht

Hier wäre es sinnvoll, die Grafikfähigkeit eines angeschlossenen Matrixdruckers besser auszunutzen. Daß der Autor sich über die Fähigkeiten verschiedener Drucker durchaus im klaren war, zeigen zwei andere Funktionen. Zum einen kann die Punktdichte des Grafikdruckes beim Epson mit ESC * variiert werden, zum anderen ist die ESC!-Funktion (Wahl der Schriftart) durch einen einfachen Parameter einzustellen.

Es hieße, einiges der Fähigkeiten einer computergestützten Textverarbeitung zu verschenken, wenn man auf einen Formatier-Modus verzichtet. Star Texter tut dies nicht. Er bietet, angewählt durch den Parameter »Trennungen« im Sammelmenü und aufgerufen aus dem Druckermenü, eine sinnvolle Funktion zum Trennen langer Worte und Sätze. Setzt man den entsprechenden Parameter, so fragt Star Texter beim Bildschirm-Formatieren (und bei der Komplett-Formatierung) in (fast) jeder Zeile nach, ob ein Wort getrennt werden soll oder nicht. Insbesondere in Verbindung mit dem ebenfalls leicht einstellbaren Blocksatz (rechter und linker Randausgleich) wird so vermieden, daß zu große Lücken in einer Textzeile entstehen. Ebenso einfach wie das Formatieren eines Textes ist auch die Wahl des Druckertyps. Dabei sind für die wohl derzeit am meisten verbreiteten Drucker (Commodore, Epson, Star) spezielle Druckmodi voreingestellt, wobei unterschieden wird, ob ein Drucker mit Hardware-Interface (Görlitz, Data Becker, Wiesemann) oder aber nur mit einem Kabel am User-Port angeschlossen ist. Betreibt man einen Drucker mit Centronics-Schnittstelle, kann jegliches Interface (bis auf das Kabel) entfallen, denn Star Texter hat einen eigenen Druckertreiber. Bei solcher Leistungsfähigkeit verwundert es kaum noch, daß alle selbstdefinierten Zeichen und die deutschen Sonderzeichen auch auf den Commodore-Druckern, die dazu normalerweise nicht in der Lage sind, wiedergegeben werden.

Star unter den Sternen

Vergleicht man die Leistungsfähigkeit und komplette Ausstattung von Star Texter mit dem Preis von 64 Mark, so dürfte es derzeit wohl kaum einen Konkurrenten geben. Zwar ist das zeilenweise Verschieben und Kopieren sicher noch zu verbessern, und auch beim Grafikdruck ist noch nicht das Optimum erreicht. Dafür bietet Star Texter aber ein umfangreiches, absturzsicheres und vor allem leicht zu bedienendes Leistungsangebot, das hauptsächlich bei den Sonderfunktionen glänzt. Frei definierbare Zeichensätze, Grafikdruck und Formatmodus sind schon fast Argument genug, um dieses Programm als gut zu bezeichnen. Der 80-Zeichen-Modus, der ohne zusätzliches Nachladen bereitsteht, dürfte für ein Programm in dieser Preisklasse wohl einmalig sein. Mit seinem anwenderfreundlichen Preis (64 Mark) und einem fast durchwegs gelungenen Konzept dringt Star Texter eindeutig in die Spitzenklasse aller Textverarbeitungsprogramme für den C64 vor.

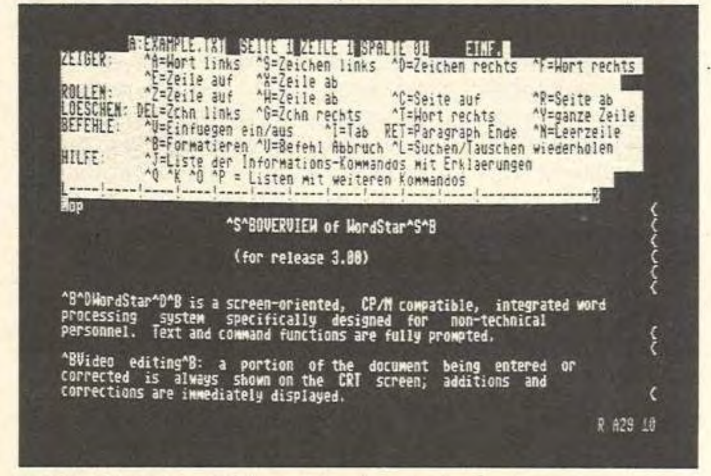


Bild 2. Wordstar: Während des Bearbeitens eines Textes werden stets die wichtigsten Kommandos angezeigt.

WordStar 128

Das älteste und wohl auch verbreitetste Textverarbeitungsprogramm für CP/M-Computer ist WordStar von MicroPro. WordStar hat eine lange Geschichte, die in die Anfangstage des Betriebssystems CP/M zurückreicht. Kurz nachdem es auf den Markt kam, hatte es sich als Standard auf dem Textverarbeitungssektor etabliert. Fast jedes folgende Textverarbeitungsprogramm versuchte, einige Funktionen oder Befehle zu übernehmen. Dennoch haben die Entwickler von WordStar ihr Produkt immer wieder verbessert; die momentan zu kaufende Version ist also immer noch aktuell.

WordStar läuft auf dem Commodore 128 unter dem Betriebssytem CP/M Plus. Dazu müssen Sie das Betriebssystem laut Computerhandbuch starten. Hier ist leider schon die erste Hürde für Anfänger zu finden. Es ist empfehlenswert, sich mit einem Einführungsbuch für dieses Betriebssystem zu befassen, um CP/M näher kennenzulernen. Für die ersten Versuche ist allerdings im Handbuch Schritt für Schritt erklärt, wie WordStar zu starten ist.

Ist das Starten erfolgreich verlaufen, offenbart sich nach kurzer Zeit das Anfangsmenü, von dem aus verschiedene Funktionen von WordStar aufzurufen sind (Bild 2). Durch Eingabe von »d« und einem Dateinamen gelangt man in den Bearbeitungsmodus, von dem aus sämtliche Funktionen zur Textmanipulation aufgerufen werden können. Zum Erfolg dieses Textverarbeitungsprogrammes trägt unter anderem die große Anzahl von Funktionen, die geboten werden, wie auch der hohe Verarbeitungsgrad des Programmes auf verschiedenen CP/M-fähigen Computern bei. Es sind sowohl Standardfunktionen enthalten (suchen, tauschen, Block bewegen, kopieren, löschen etc.), als auch eine Reihe ausgefallener Möglichkeiten. So ist zum Beispiel ein Befehl vorhanden, die Seitennumerierung abhängig von der Seitenzahl am linken oder rechten Rand erscheinen zu lassen (das ist für Druckvorlagen sehr angenehm). Ebenso wird eine komfortable Behandlung von Textbausteinen angeboten. Sogenannte Textbausteine sind einzelne, immer wiederkehrende Phrasen oder Abschnitte, wie zum Beispiel ein Briefkopf oder Angebotstexte. Diese können einzeln erstellt und gespeichert werden. So ist es im Extremfall möglich, daß ein ganzer Brief nur aus Textbausteinen zusammengesetzt wird. Einzig die Adresse müßte noch von Hand eingesetzt werden. Aber auch dafür existieren Methoden, die später beschrieben werden.

Aufgerufen werden diese Befehle durch ein bis drei Tastendrücke. Hier liegt die zweite Hemmschwelle, die allerdings nicht nur für Computerneulinge gilt. Bevor Sie nicht diese Befehlskombination in etwa beherrschen, artet die Arbeit mit WortStar in Handbuchblättern aus. Um diesem Zustand abzuhelfen, wurden zwei Vorkehrungen getroffen. Das Handbuch bietet eine Übersichtskarte mit allen Befehlen und deren Kurzbeschreibung, das Programm selbst zeigt eine Tabelle der momentan aufrufbaren Funktionen an. Bei Befehlen, die durch Drücken verschiedener Tasten aufgerufen werden, wird kurz nach dem Drücken der ersten Taste die zu diesem Befehl gehörige Tabelle mit allen weiteren möglichen Tastendrucken angezeigt, falls nicht sofort weitere Eingaben erfolgen. Für erfahrene WordStar-Benutzer gibt es die Möglichkeit, das Anzeigen der Tabellen zu unterdrücken.

Schon diese kurze Übersicht vermittelt einen Eindruck der Vielseitigkeit von WordStar. Das Einsatzgebiet liegt vorrangig im professionellen Bereich. Es ist ohne weiteres möglich, geschäftlichen Schriftverkehr mit diesem Programm abzuwickeln. Unterstützt wird gerade der kommerzielle Einsatzbereich durch den mitgelieferten »MIX-Druck«, im Original Mailmerge genannt. Dieses Zusatzprogramm bietet die Möglichkeit, Adressen, Anschriften oder ganze Textteile beim Druck einer anderen Quelle zu entnehmen. Die andere Quelle wird in den meisten Fällen eine Datei auf Diskette sein. Die Datei kann entweder mit WordStar selbst erstellt worden sein oder von anderen Programmen kommen (zum Beispiel von Multiplan 128 oder dBase II). Mit Hilfe von Mailmerge sind sogenannte Serienbriefe möglich. Das sind gleichlautende Briefe, die an verschiedene Empfänger geschickt werden. Dadurch, daß bei jedem Brief eine individuelle Adresse, Anschrift, Anrede oder andere Informationen eingesetzt werden, sehen die Briefe nicht nach Serienbriefen aus.

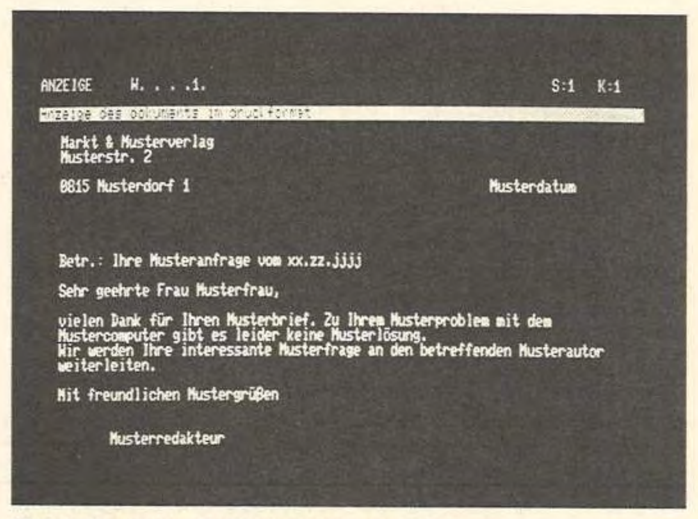


Bild 3. So sieht ein Brief mit Superscript aus

Insgesamt betrachtet besticht WordStar durch seine vielfältigen Möglichkeiten, die allerdings, wie schon gesagt, durch die für Anfänger sehr schwer zu durchschauende Bedienung erkauft werden muß. Ist man aber einmal eingearbeitet, so möchte man vieles nicht mehr missen.

Wer schon einmal mit WordStar auf einem anderen CP/M-Computer oder IBM-PC geschrieben hat, wird allerdings beim C128 etwas enttäuscht sein. Die Verarbeitungsgeschwindigkeit läßt doch zu wünschen übrig. Schuld daran ist jedoch nicht WordStar, sondern vielmehr das CP/M des C128.

Hier zeigt sich ziemlich deutlich, daß das CP/M nicht das bevorzugte Betriebssystem des C128 ist. Sehr schnelle Programme können viel eher im eigentlichen C-128-Modus entstehen. Ein Beispiel dafür ist Protext.

Superscript

Superscript entstand in der Programmschmiede Precision Software, die auch so bekannte Produkte wie »Easyscript« und »Superbase« entworfen hat. Bei der Entwicklung von Superscript wurde besonders darauf geachtet, gerade den Speichergewinn beim C128 so gut wie möglich auszunutzen. So verfügt man über zwei Textspeicher von 58 und 17 KByte, zwischen denen man Texte frei hin- und herschieben kann. Das dürfte für normalen Einsatz mehr als genügen, zumal man auch mehrere Textdateien miteinander verbinden kann. Bei Superscript kann man sich aussuchen, in welchem Modus man arbeiten will: 40 oder 80 Zeichen pro Zeile. Wem 80 Zeichen zum Editieren immer noch nicht ausreichen, der kann eine neue Zeilenbreite beliebig festlegen, wobei der Text dann horizontal hin- und hergeschoben wird. Ein Übergang zum Datenbankprogramm »Superbase 128« aus gleichem Hause ist ebenfalls möglich, wobei allerdings alle bestehenden Daten im Speicher verloren gehen.

Eigenwillige Menütechnik

Bei fast allen Textverarbeitungsprogrammen werden einzelne Befehlsaufrufe über Sondertasten wie ESC, CTRL oder

andere eingeleitet, gefolgt von einer Buchstabentaste, die dann den eigentlichen Befehl aktiviert. Bei Superscript dagegen werden selbst die einfachsten Befehle wie »Tabulator setzen« über ein Hauptmenü und mehrere Untermenüs angewählt. So gibt es insgesamt mehr als 25 Menüs, über die sämtliche Funktionen des Programms ansprechbar sind. Das jeweilige Menü wird dabei immer in der ersten Zeile angezeigt, wobei man dann mittels Cursortasten oder Anfangsbuchstaben der aufzurufenden Funktion auswählen kann. Dabei können Untermenüs weitere Untermenüs enthalten. Diese auf den ersten Blick sehr umständlich und überholt wirkende Methode wird allerdings dadurch entscheidend aufgewertet, daß man über die ESC- und die Commodore-Taste mit Hilfe der Buchstaben weit über 50 Befehlsketten definieren kann. So kann sich jeder selbst seine spezifischen Funktionsaufrufe zusammenbasteln, und damit die langwierige Menütechnik vollständig umgehen.

Die festgelegten Funktionen sollten dabei in eine Datei mit dem Namen »Defaults« eingetragen werden, die immer beim Laden des Programmes mit eingelesen wird. Dort kann man auch die Bildschirmfarben individuell einstellen. Weiterhin findet sich dort ein Verweis auf eine Datei, in der die speziellen Druckerparameter untergebracht sind. So kann man das Programm an nahezu jeden Drucker anpassen. Für die wichtigsten Drucker sind entsprechende Dateien schon eingerichtet.

Gute Grundfunktionen

Selbstverständlich sind Standardbefehle wie Einfügen, Löschen, Kopieren und Verschieben im Programm realisiert.

Überdurchschnittlich gut ausgefallen sind die Such- und Ersetzfunktion. Sie erlauben Vorwärts- und Rückwärtsbetrieb sowie das Quittieren bei jedem gefundenen Wort. Somit lassen sich einzelne Stellen beim Ersetzen überspringen. Mit Hilfe mehrerer Rechenfunktionen läßt sich im gewissen Rahmen auch Tabellenkalkulation durchführen.

Floppybehandlung

Komfortabel gelöst ist das Laden von Texten. Man kann hier direkt aus einem Disketten-Inhaltsverzeichnis laden, ohne den Dateinamen eingeben zu müssen. Da Superscript sequentielle Dateien erzeugt, gibt es auch hier keine Schwierigkeiten mit der Verarbeitung von Fremddateien. Ebenfalls sehr nützlich ist das Umleiten eines Ausdrucks in eine Datei auf Diskette. Floppybefehle und ein vollständiges Inhaltsverzeichnis sind ebenfalls kein Problem.

Erst schreiben, dann drucken

Da der Text beim Eintippen noch nicht so aussieht, wie er dann später gedruckt wird, muß man mit entsprechenden Textbefehlen die Formatierung festlegen. Auch gibt es die Möglichkeit, sich den Text vor dem eigentlichen Ausdruck nur auf dem Bildschirm zeigen zu lassen. Dabei werden auch Sonderformen wie Unterstreichen, Fett- und Doppelschrift oder Breitdruck mit umgesetzt. Wurde die Druckbreite auf mehr als 80 Zeichen eingestellt, kann man mittels horizontalem Verschieben auch die rechte Seite des Textes einsehen. Überzeugend unterstützt wird die Erstellung von Serienbriefen mit Dateneinzug aus SEQ-Dateien. Hierbei ist es sogar möglich, einzelne Datensätze je nach Inhalt zu drucken oder auszulassen (vergleichende Abfrage). Superscript 128 unterstützt beim Ausdruck den seriellen Bus, die RS232-Schnittstelle sowie ein am User-Port eingerichtetes Centronics-Interface. Damit dürfte es auch bei ausgefallenen Druckern oder Schreibmaschinen keine Probleme geben.

Protext 128 – Bewährtes erweitert

Viele neue Programme für den C128 sind umgebaute und erweiterte Programme, die vom C64 kommen, so auch Protext 128. Man muß allerdings sagen, daß die ältere Version für den C64 (mit 80-Zeichen-Karte) schon über eine Menge guter Eigenschaften verfügte. Diese Grundeigenschaften wurden aber bei der Umsetzung auf den C128 noch einmal kräftig nach oben erweitert.

Gute Grundfunktionen

Selbstverständlich sind die üblichen Grundmanipulationen wie Einfügen, Löschen, Verschieben und Kopieren einzelner Texteile vorhanden. Überdurchschnittlich gut gelöst sind die sehr wichtigen Such- und Ersetzfunktionen für einzelne Textstellen. So kann man hier mit dem von der Floppy her bekannten Jokerzeichen »?« arbeiten und damit im zu suchenden Wort ein beliebiges Zeichen markieren.

Nützlich ist auch die Möglichkeit, beim Ersetzen einzelne Vorkommen des Wortes gezielt auslassen zu können. Das Programm arbeitet ausschließlich im 80-Zeichen-Modus, so daß ein entsprechender Farb- oder Monochrom-Monitor zwingend erforderlich ist. Wem das immer noch zu wenig ist, der kann auf 120 Zeichen pro Zeile umschalten, wobei dann horizontal gescrollt wird. Das ist aber nur für die Verarbeitung breiter Tabellen wichtig, da der Text erst beim Ausdruck endgültig formatiert wird. Solche Tabellenverarbeitung wird weiterhin durch einen speziellen Spaltenmodus, eine Sortierroutine und umfangreiche Rechenfunktionen sehr gut unterstützt. Die vorhandenen zwei Textspeicher können dabei je 250 Zeilen Text aufnehmen, womit man dann auf insgesamt 60 KByte zugreifen kann (Bild 4).



Bild 4. Protext 128 schreibt Ihre Mahnung

Spezialfunktionen mit Pfiff

Völlig aus dem Rahmen üblicher Textverarbeitungsprogramme fallen aber einige der Sonderfunktionen, über die das Programm verfügt. So kann man beispielsweise den Bildschirm in zwei Textfenster aufteilen und auf diese Weise zwei verschiedene Texteile miteinander vergleichen. Dabei kann natürlich immer nur eines der beiden Fenster bearbeitet werden. Ungewöhnlich ist die eingebaute Silbentrennung beim Ausdruck, die alle Regeln und die wichtigsten Ausnahmen beherrscht. Ebenfalls angenehm ist die Möglichkeit, einen Text mittels einer eingebauten Korrekturroutine zu überprüfen. Es steht dabei eine Grundbibliothek von 20000 Wörtern zur Verfügung, die man selbst weiter ausbauen kann.

Auch die HELP-Funktion besticht durch ihre Reichhaltigkeit. Zu jedem Befehl ist ein entsprechender Erklärungstext von Diskette verfügbar, man muß nur nach dem Drücken der HELP-Taste die entsprechende Tastenkombination eingeben. Mit der Definiton einzelner, sogenannter Makros lassen sich ganze Befehlsketten bilden, die man auch selbstaufrufend definieren kann. Weiterhin lassen sich Jobdateien einrichten, mit denen man einen Druckauftrag, der aus mehreren

Einzeltexten in frei wählbarer Reihenfolge besteht, erstellen kann. Einer erst in jüngster Zeit immer beliebter werdenden Tätigkeit wurde auch in diesem Programm Rechnung getragen: der Datenfernübertragung. Nach der Einstellung der Parameter wie Baudrate, Datenbits, Paritätsprüfung und Textpuffer (an/aus) kommt man direkt in einen Terminalmodus und kann dann mit der Textübertragung per Telefon beginnen. Besonders interessant ist ein speziell gegen Leitungsfehler abgesicherter Übertragungsmodus, mit dem zwei Besitzer von Protext Daten austauschen können.

Der Floppyzugriff

Gut gelungen sind auch die Lade- und Speicherroutinen. So kann man beispielsweise den Namen des zu ladenden Textfiles aus dem Inhaltsverzeichnis der Diskette aussuchen und muß ihn so nicht unbedingt selbst eingeben. Besonders interessant ist die Speicherung von Texten mittels eines Paßwortes, das zum korrekten Laden eingetippt werden muß. So kann man seine Daten wirkungsvoll vor fremdem Zugriff sichern. Nachteilig ist hier nur zu vermerken, daß es keine Möglichkeit gibt, sich ein Directory mit allen Daten wie Dateiname, Dateityp und Länge in Blocks anzeigen zu lassen. Das kann zu Verwirrungen führen.

SEQ-Dateien können mit einer eigenen Funktion in das spezielle Protext-Format umgewandelt werden. Auch das Umwandeln von Protext-Text in SEQ-Dateien ist mit dieser Funktion möglich.

Vielseitig im Druck

Für Protext existiert eine Vielzahl von Druckertreibern für alle möglichen Kombinationen. Zusätzlich zu den schon bestehenden kann man sich einen Drucktreiber mittels einer komfortablen Unterfunktion neu erstellen. Der Ausdruck kann dabei beliebig auf den seriellen Bus, die RS232-Schnittstelle oder ein am User-Port eingerichtetes Centronics-Interface an den Drucker geleitet werden. Im letzten Fall ist dazu nur ein einfaches Verbindungskabel nötig. Die Ansteuerungssoftware ist bereits in Protext enthalten.

Fazit

Protext stellt eine gelungene Anpassung und Erweiterung eines bewährten Programms an einen neuen Computer dar. Dabei besticht dieses Programm vor allem durch seine gut durchdachten Extrafunktionen; wie Terminalmodus, Textfenster, Silbentrennung, Textkorrektur und Makrobefehle sowie durch seinen konkurrenzlos günstigen Preis von 89 Mark. Mit Protext 128 wurde ein gelungenes Programm vorgestellt, mit dem man auf dem C128 professionell arbeiten kann und das sich hinter so manchem Textverarbeitungsprogramm für einen PC nicht zu verstecken braucht.



Multiplan 128

Im Bereich Kalkulationsprogramme sticht ein Programm hervor: Es heißt Multiplan und ist von der Firma Microsoft geschrieben. Die Möglichkeiten, die sich mit Multiplan dem Anwender auftun, sind fast unübersehbar. Einen ersten Überblick gibt das sehr umfangreiche Handbuch, welches außer der Befehlsübersicht noch Anwendungsbeispiele enthält. Multiplan arbeitet nach folgendem Prinzip: Dem Benutzer wird ein Rechenfeld zur Verfügung gestellt, von dem er immer einen wählbaren Ausschnitt auf dem Bildschirm anzeigen kann. Das Rechenfeld ist in einzelne Felder unterteilt, die über ihre Zeilen- und Spaltennummer oder über frei wählbare Namen angesprochen werden. Diese Felder können Texte, Werte oder Formeln enthalten. Dabei können Formeln wiederum Formeln oder Werte, die in anderen Feldern stehen, mit einbeziehen. Auf diese Weise sind sehr komplexe Verknüpfungen möglich. Wird der Inhalt eines Feldes verändert, so rechnet Multiplan automatisch alle Felder neu durch. Dadurch sind auf einfache Weise sogenannte »Was-ist-Wenn«-Betrachtungen möglich, die im Geschäftsbereich wichtig sind. So ist es leicht auszurechnen, ob es zum Beispiel günstiger ist, drei Arbeiter einzustellen und zwei Maschinen zu kaufen, oder einen Arbeiter einzustellen, dafür aber drei Maschinen anzuschaffen.

Multiplan erlaubt es, eine Rechnung mit Lieferschein zu erstellen, gleich zu adressieren und auszudrucken. Automatisch wird der aktuelle Lagerbestand ermittelt und auf eventuell erforderliche Nachbestellungen hingewiesen. Sogar Serienbriefe und Adreßdateien lassen sich mit Multiplan verwirklichen. Aber der Anwendungsfall muß nicht so kompliziert sein. Für jemand, der seine Steuerabrechnung erledigen will, ein Haushaltsbuch führt oder einfach das Zahlenrätsel aus der Fernsehzeitung lösen möchte, ist Multiplan ebenso geeignet. Wissenschaftlern, die eine Reihe von mathematischen Funktionen benötigen, ist ebenso gedient. Es sind sogar Statistikfunktionen enthalten. Es ist einfach zu sehen, daß dem Anwender mit Multiplan kaum Beschränkungen auferlegt werden, was die Vielseitigkeit des Programms ausmacht. Sollte man gerade wegen der großen Anzahl von Möglichkeiten einmal nicht mehr wissen, was ein bestimmter Befehl bedeutet, stellt Multiplan eine sehr umfangreiche HELP-Funktion zur Verfügung, die das Handbuch fast überflüssig macht.

Für welche Aufgabe Multiplan geeignet ist, hängt allein von der Programmierung des Rechenfeldes ab. Darin liegt ein großer Vorteil des Konzepts: Das Programm bleibt dynamisch. Wachsen die Anforderungen oder ändert sich die Aufgabenstellung, so wird einfach die Programmierung geändert. Für die, die bei dem Wort »programmieren« einen Rückzieher machen, sei gesagt, daß sich die Programmiertätigkeit auf das Einsetzen von Formeln in Felder beschränkt, was relativ einfach zu erlernen ist. Ist ein Rechenblatt fertig erstellt, können Sie es auf Diskette speichern. Sollte das Ganze immer noch zuviel werden, so gibt es vorgefertigte Rechenblätter, die auf einen bestimmten Anwendungsfall zugeschnitten sind. Dabei besteht natürlich immer noch die Möglichkeit, einzelne Felder zu ändern, um das Rechenblatt an den ganz persönlichen Anwendungsfall anzupassen.

Die Ausgabe von Daten mit Multiplan muß nicht unbedingt auf den Drucker erfolgen, sie kann auch auf Diskette umgeleitet werden. Dadurch ergibt sich die Möglichkeit, die Daten mit einem Textverarbeitungsprogramm zu bearbeiten. Dies erleichtert es dem Anwender, die Daten zu kommentieren, in eine bestehende Tabelle zu übernehmen oder einfach in

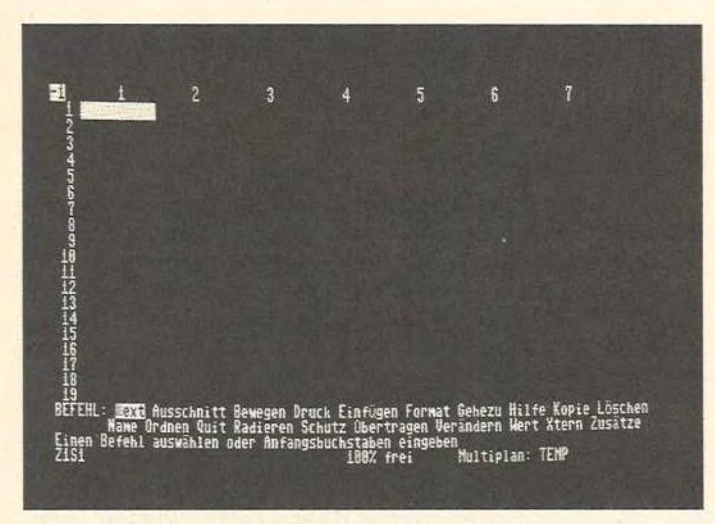


Bild 5. Das leere Arbeitsblatt von Multiplan 128

einen Text einzufügen. Das Erstellen von Geschäftsberichten und Bilanzen wird dadurch sehr vereinfacht. Alles in allem ist Multiplan eine exzellente Tabellenkalkulation. Neben der Dateiverwaltung eröffnet das Programm neue Dimensionen in Zusammenarbeit mit einem Textverarbeitungsprogramm, wie etwa das schon beschriebene WordStar.

Leider haftet Multiplan, wie auch allen hier beschriebenen Programmen für den Commodore 128, der gleiche Mangel an: Die Bedienung dieser Programme ist relativ kompliziert, weil sie recht vielseitig ist. Daher erfordert ihre Bedienung etwas Wissen über die Arbeitsweise von Computern. Dennoch möchte ich absoluten Computerneulingen nicht vom Kauf dieser Programme abraten. Die Vorteile, die diese Programme bieten, sind einfach zu groß, um darauf zu verzichten. Sie müssen nur einige Stunden Einarbeitungszeit einkalkulieren, die später durch den Nutzen mehr als wettgemacht werden. Es gibt einige Bücher, die auf einfache Weise in die Bedienung der Programme einführen und so Starthilfe und weitere Informationen geben.

Die hier besprochene Version für den C128 läuft unter CP/M und hat deutsche Umlaute, 80 Zeichen pro Zeile und bietet die Möglichkeit, Drucker mit Centronics-Schnittstelle anzusteuern (Bild 5).

Superbase 64

Grundsätzlich kann mit Superbase 64 ohne irgendwelche Programmierkenntnisse gearbeitet werden. Wer jedoch tiefer einsteigen will und sich mit den »normalen« Funktionen noch nicht zufriedengibt, bekommt im dritten Teil des Handbuches genügend Tips und Tricks zum Experimentieren.

Von den Anwendungsmöglichkeiten zerfällt das Softwarepaket in zwei Bereiche:

1. Datenbank-Management-System
2. Kalkulation und Fakturierung

Um mit Superbase 64 arbeiten zu können, braucht man einen Commodore 64 und ein Diskettenlaufwerk VC 1541. Im Gegensatz zu manch anderem Dateiverwaltungsprogramm genügt hierzu wirklich ein einziges Diskettenlaufwerk, da das Programm komplett geladen wird, beziehungsweise die erforderlichen externen Teile mit auf die Datendiskette kopiert werden. Das erfordert zwar etwas mehr Vorarbeit, macht aber den späteren Arbeitsablauf angenehm und erfordert keine Discjockey-Fähigkeiten. Da grundsätzlich alle Datenbankabfragen und -auswertungen sowohl über Drucker als auch über Bildschirm möglich sind, wird der Drucker nicht unbedingt benötigt, um den vollen Funktionsumfang nutzen zu können.

Grundsätzlich kann eine Superbase-Datenbank bis zu 15 Einzeldateien umfassen. Ein Datensatz kann maximal 1108 Zeichen lang sein und auf 4 Bildschirmseiten verteilt werden. Pro Satz sind maximal 127 Felder inklusive Schlüsselfeld möglich, wobei das Schlüsselfeld maximal 30 Stellen lang sein kann; normale Textfelder können bis zu 255 Zeichen beinhalten und sich somit auch über mehrere Zeilen erstrecken. Nachdem man sich im Formatierungsmodus befindet, wird links oben der jeweilige Modus angezeigt, was eine angenehme Orientierungshilfe darstellt.

Auf dem übrigen leeren Bildschirm werden die einzelnen Maskenfelder definiert. Dies läuft im einzelnen folgendermaßen ab:

- Feldnamen eingeben.
- Cursor auf Feldanfang positionieren.
- F1-Taste drücken und Buchstaben für Felddefinition (zum Beispiel »K«=Key, »T«=Text, »D«=Datum) eingeben.
- Feld mit Cursor durchlaufen, bis der Stellenzähler rechts oben die gewünschte Feldlänge anzeigt.
- Feldende mit RETURN-Taste markieren.

Alle Felder sind nun am Feldanfang mit einem kleinen ausgefüllten Quadrat und am Feldende mit einem größeren schraffierten Quadrat gekennzeichnet. Mit F1-Taste und »I« können Felder invertiert dargestellt werden, mit F1-Taste und »S« wird der gesamte Bildschirm invertiert. Nachträgliche Änderungen sind etwas umständlicher durchzuführen: Mit F1-Taste und »E« (=Error) wird nur der Feldname gelöscht; um das Feld selber zu löschen, muß der Cursor genau auf den Feldanfang positioniert und nochmals F1-Taste und »E« eingegeben werden.

Wenn die Maske endgültig definiert ist, wird sie mit der F1-Taste und der RUN/STOP-Taste abgespeichert. Gleichzeitig werden die bisherigen Feldbegrenzungszeichen in »<« und »>« umgesetzt. Sehr angenehm in der Bedienung ist es, daß bei numerischen Eingabefeldern automatisch die Buchstaben auf der Tastatur gesperrt sind, was Eingabefehler verhindert.

Von nun an wird das Arbeiten mit Superbase immer mehr zum Vergnügen. Mit RETURN kann man einfach in das Hauptmenü umsteigen. Die jeweils gewünschte Funktion kann wahlweise über die entsprechende Funktionstaste oder die Eingabe in der Kommandozeile ausgelöst werden. Mit der F8-Taste kann aus dem Hauptmenü I jederzeit eine HELP-Funktion aufgerufen werden. Die Beschreibung der gewünschten Funktion kann hiermit aufgerufen werden. Außerdem können auch eigene HELP-Masken definiert werden.

Daten eingeben:

Über Hauptmenü I und der F1-Taste wird die ENTER-Funktion gestartet. Sofort erscheint eine leere Bildschirmmaske und links oben »Mode Entry«. Mit RETURN kommt man an den Anfang des nächsten Feldes, mit HOME an den Maskenanfang. Drückt man nach dem Ausfüllen des letzten Feldes nochmals RETURN, so wird der Satz sofort abgespeichert. Werden nicht alle Felder ausgefüllt, so kann das Abspeichern schon vorher mit SHIFT/RETURN ausgelöst werden. Mit der SPACE-Taste bekommt man das nächste Leerbild. Das Duplizieren von Sätzen ist über »SELECT« und »A« (=add) möglich. Die Dateneingabe ist schnell und denkbar einfach.

Hier bietet Superbase 64 nahezu unbegrenzte Möglichkeiten. Der schnellste Zugriff erfolgt über das Schlüsselfeld (Key). Die Möglichkeiten der »SELECT«-Funktion reichen vom beliebigen Blättern in der Datenbank bis zu kompliziertesten UND- beziehungsweise ODER-Verknüpfungen, verbunden mit Vergleichsoperationen wie größer, kleiner oder ungleich (>, <, #). Suchen mit Joker ist genauso problemlos möglich wie Suchen über Matchcode.

Die aus der gesamten Datei ausgewählten Sätze können nacheinander am Bildschirm angezeigt werden. Sehr ange-

nehm ist es auch, daß man jederzeit auf nicht ausgewählte »Nachbar-Sätze« springen kann. Die ausgefilterten Ergebnisse zu einer Abfrage können auch in einer Hilfsdatei abgelegt werden. Die Selektionsmöglichkeiten von Superbase 64 lassen absolut keine Wünsche unbefriedigt. Durch optimales Zusammenwirken der Funktionen »SELECT«, »SORT« und »OUTPUT« lassen sich beliebige Datenbankauswertungen in übersichtlicher Form darstellen. Die Ausgabe kann wahlweise über Bildschirm oder Drucker erfolgen. Wer nicht unbedingt ein schriftliches Protokoll benötigt, kann auch ohne teuren Drucker fast alle Vorteile von Superbase 64 voll nutzen.

Daß Superbase 64 auch über einen leistungsfähigen Listenprogrammgenerator (REPORT-Funktion) verfügt, gehört nach den bisher aufgezeigten Funktionen einfach zu den Selbstverständlichkeiten.

Resümee bei der Arbeit mit Superbase 64 als Datenbanksystem: Hat man die nicht immer ganz einfachen Vorleistungen zur Erstellung der Datenbank inklusive Maskendefinition hinter sich gebracht, so macht die Dateneingabe und -pflege Vergnügen. Die Abfrage- und Auswertungsmöglichkeiten sind nahezu unbegrenzt, das heißt, man kann zu Recht von einer »SUPER«-Datenbank sprechen.

Die vielfältigen Funktionen von Superbase 64 als Datenbanksystem waren schon sehr beeindruckend. Darüber hinaus enthält dieses Programm jedoch noch Kalkulationsmöglichkeiten, wie sie sonst oft nur von einem separaten Programm abgedeckt werden. Damit werden auch Anwendungsgebiete wie Fakturierung und Verkaufsstatistik erschlossen.

Verkaufsstatistik:

Bedient man sich der Funktion »OUTPUT«, so kann man sich mit der Angabe ALL eine komprimierte Übersicht über alle verkauften Artikel, Stückzahlen und Beträge ausgeben lassen. Natürlich könnte diese Übersicht zum Beispiel auch nach Artikelbezeichnung sortiert werden. Eine Verkaufsstatistik auf Bildschirm oder Liste ist also kein Problem.

Dienstprogramme:

Der gute Eindruck von einem soliden und leistungsfähigen Software-Produkt wird noch abgerundet durch einige nützliche Dienstprogramme. Unter anderem kann mit Catalog jederzeit der Superbase-interne Katalog angegeben werden, beziehungsweise mit Directory der Inhalt der gesamten Diskette. Mit Backup können schnell Sicherungskopien der Datendiskette erstellt werden. Besonders muß noch auf die Funktionen Import/Export hingewiesen werden:

- Mit Import können Daten von externen Programmen eingelesen und mit Superbase-Dateien zusammen weiterverarbeitet werden.
- Mit Export können Daten aus Superbase-Dateien als sequentielle Datei ausgegeben werden, die dann von anderen, zum Beispiel Basic-Programmen, weiterverarbeitet werden können.

Abschließende Beurteilung:

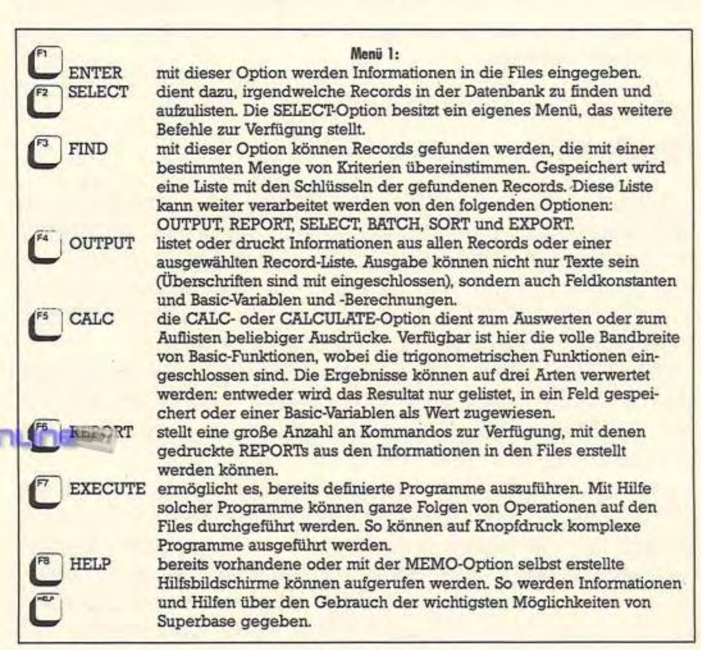
Für zirka 300 Mark bekommt der Käufer ein sehr bedienungsfreundliches Software-Paket, das so viele Möglichkeiten bietet, wie man sie sonst oft nur mit mehreren Einzelprogrammen geboten bekommt. Die Funktionen der »SUPER«-Datenbank sind beeindruckend. Die zusätzlichen Kalkulationsmöglichkeiten eröffnen ein breites Anwendungsgebiet, das weit über reine Datenverwaltungen hinausgeht. Besonders angenehm sind auch die komprimierten Auswertungen wahlweise auf Bildschirm oder Drucker. So bleibt mit Superbase 64 kaum ein Wunsch unerfüllt.

Superbase für den C128

Geliefert wird von Precision Software eine Programmdiskette und das Handbuch zum Programm. Der erste Teil des Handbuchs ist als Tutorium aufgebaut. Hier werden dem Neuling anhand kleiner Probleme Anwendungen von Superbase portionsweise zum Nachmachen vorgesetzt. Der zweite Teil des Handbuchs beschreibt die Menüs und die Befehle des Systems systematisch. Die Superbase-Diskette wird in das Laufwerk geschoben, der C128 eingeschaltet und schon geht es los: Superbase wird geladen.

Nach 70 Sekunden ist das Programm im C128 und der Benutzer kann wählen: Soll eine neue Daten-Diskette erstellt werden oder wird eine formatierte Diskette verwendet?

Mit dem Befehlsvorrat kann man für die Anforderungen eines Anwenders vom Definieren einer Bildschirmmaske über sortierte Datenbank-Auswertungen bis hin zu Kalkulation und Fakturierung relativ problemlos Programmlösungen finden. Eine Kunden- oder Rechnungsdatei ist schnell aufgebaut. Eine Verkaufsstatistik kann von einem geübten Superbase-Anwender in kurzer Zeit programmiert werden.



Menü 1:

Taste	Option	Beschreibung
F1	ENTER	mit dieser Option werden Informationen in die Files eingegeben.
F2	SELECT	dient dazu, irgendwelche Records in der Datenbank zu finden und aufzulisten. Die SELECT-Option besitzt ein eigenes Menü, das weitere Befehle zur Verfügung stellt.
F3	FIND	mit dieser Option können Records gefunden werden, die mit einer bestimmten Menge von Kriterien übereinstimmen. Gespeichert wird eine Liste mit den Schlüsseln der gefundenen Records. Diese Liste kann weiter verarbeitet werden von den folgenden Optionen: OUTPUT, REPORT, SELECT, BATCH, SORT und EXPORT.
F4	OUTPUT	listet oder druckt Informationen aus allen Records oder einer ausgewählten Record-Liste. Ausgabe können nicht nur Texte sein (Überschriften sind mit eingeschlossen), sondern auch Feldkonstanten und Basic-Variablen und -Berechnungen.
F5	CALC	die CALC- oder CALCULATE-Option dient zum Auswerten oder zum Auflisten beliebiger Ausdrücke. Verfügbar ist hier die volle Bandbreite von Basic-Funktionen, wobei die trigonometrischen Funktionen eingeschlossen sind. Die Ergebnisse können auf drei Arten verwertet werden: entweder wird das Resultat nur gelistet, in ein Feld gespeichert oder einer Basic-Variablen als Wert zugewiesen.
F6	REPORT	stellt eine große Anzahl an Kommandos zur Verfügung, mit denen gedruckte REPORTs aus den Informationen in den Files erstellt werden können.
F7	EXECUTE	ermöglicht es, bereits definierte Programme auszuführen. Mit Hilfe solcher Programme können ganze Folgen von Operationen auf den Files durchgeführt werden. So können auf Knopfdruck komplexe Programme ausgeführt werden.
F8, HELP	HELP	bereits vorhandene oder mit der MEMO-Option selbst erstellte Hilfsbildschirme können aufgerufen werden. So werden Informationen und Hilfen über den Gebrauch der wichtigsten Möglichkeiten von Superbase gegeben.

Tabelle 1. Die Optionen des ersten Superbase-Menüs (Menü 1)

Menü 2:

Taste	Option	Beschreibung
F1	FILE	erzeugt einen neuen File oder ändert den aktuellen File in der Datenbank.
F2	FORMAT	diese Option wird (1) automatisch von FILE aufgerufen, um das Layout eines neuen Files auf dem Bildschirm festzulegen, oder (2) vom Menü, um das Bildschirm-Layout des aktuellen Files zu ändern.
F3	BATCH	Berechnungen werden aufgrund von Informationen aus Records des aktuellen Files durchgeführt.
F4	SORT	mit Hilfe dieser Option können alle Records oder eine ausgewählte Record-Liste aufgrund von Feldern, die nicht Schlüsselfeld sind, sortiert werden. Das Resultat der SORT-Operation ist ein File, der nur aus den Record-Schlüsseln in der sortierten Reihenfolge besteht.
F5	PROG	erzeugt und speichert Programme. Die dazu verwendete Sprache ist Basic, das um alle Superbase-Kommandos erweitert wurde.
F6	MAINTAIN	stellt ein weiteres Menü zur Verfügung mit Optionen, die verschiedene Utilities auf den Files bereitstellen, unter anderen auch den EXPORT und IMPORT von Daten von anderen Programmen und an diese
F7	MEMO	diese Option erlaubt es, Bildschirme selbst zu erstellen, auf die später von einem selbst oder von anderen Superbase-Benutzern zugegriffen werden kann. Insbesondere können so Hilfsbildschirme entworfen oder abgeändert werden.
F8, HELP	HELP	gibt auf einem Hilfsbildschirm Informationen und Erinnerungshilfen für die wichtigsten Befehle von Superbase. Auch Hilfsbildschirme, die mit der MEMO-Option selbst erstellt wurden, können mit HELP aufgerufen werden.

Tabelle 2. Die Optionen des zweiten Superbase-Menüs (Menü 2)

Das Handbuch ist sehr übersichtlich aufgebaut. Im ersten Teil – dem sogenannten Tutorium – werden die Möglichkeiten von Superbase in einzelnen Schritten beschrieben. Die Erläuterung beginnt bei der Tastatur und führt hin bis zur komplexen Fakturierung. Der zweite Teil des Handbuchs orientiert sich an den Bildschirm-Menüs und listet die Befehle systematisch. Zu jedem Kommando gibt es ausführliche, sehr gut verständliche Erklärungen.

Superbase und dBase II bauen auf unterschiedlichen Konzepten auf, sind aber ansonsten gleichwertig. Für den Anfänger scheint dBase II wesentlich besser geeignet, weil leichter erlernbar zu sein. Dafür hat der Anwender mit nur einem Laufwerk an dBase II keine Freude, weil er neben dem Programm kaum noch Daten unterbringen kann (die Programmdiskette muß im Laufwerk bleiben, weil dBase II immer wieder nachlädt). Für den Anwender, der Zeit hat, sich einzuarbeiten, der häufige und komplexe Berechnungen mit seinen Daten durchführen will oder den Wert auf Schönheit beim Design der Daten legt – für den ist Superbase optimal geeignet. Superbase kann auch mit nur einem Floppy-Laufwerk betrieben werden.

StarDatei

Bekannt und berühmt ist das Textverarbeitungsprogramm »StarTexter«. Wer damit schon einmal gearbeitet hat, kommt auf Anhieb mit »StarDatei« gut zurecht. Viele Operationen sind ähnlich. Es gibt hier genauso einen Control-Modus, in den man mit der »CTRL-Taste« gelangt, der auch die Menüauswahl (es stehen drei zur Verfügung) steuert.

Gleich nach dem Laden des Hauptprogrammes gelangt man in das Eingabefeld von »StarDatei«. Eine Maskendefinition, wie bei anderen Dateiverwaltungsprogrammen, wird hier nicht angewendet.

Jeder Datensatz hat immer ein Eingabefeld von 19 mal 40 Zeichen zur Verfügung. Um immer wiederkehrende Bezeichnungen wie »Adresse:« oder ähnliche nicht bei jedem Datensatz neu schreiben zu müssen, kann man sich eine komplette Seite »merken« lassen und bei Bedarf jederzeit auf den Bildschirm zurückholen. Es gibt keine festen Datenfelder mit einer erzwungenen Eingabe (Bild 6).

Dieses System hat aber auch seine Nachteile. Es fehlen Hilfen wie TAP oder POS, um schnell eine bestimmte Bildschirmposition zu erreichen. Ebenso wird bei jedem Datensatz die gesamte Information, also auch die Feldbezeichnungen wie »Name, Adresse oder Telefonnummer«, mitgespeichert. Der Platzverbrauch auf der Diskette ist daher sehr groß. Jeder Datensatz hat eine Titelzeile, die ein schnelles Auffinden ermöglicht. Nach ihr werden Datensätze alphabetisch sortiert. Da die Daten in einem besonderen Algorithmus gespeichert sind, wird die Diskette entsprechend formatiert. Jede Diskette darf nur eine Datei enthalten. Dieses ermöglicht dann in wenigen Sekunden ein Wiederauffinden einzelner Datensätze. Praktisch ist es, daß man jederzeit die Diskette und damit die Datei wechseln kann. Das Programm erkennt einen Diskettenwechsel automatisch und reagiert dann entsprechend. Einen guten Eindruck macht »StarDatei« beim Suchen und Finden von Datensätzen. Mit den von der Floppy bekannten Jokerzeichen »*« und »?« lassen sich die Kopfzeilen sekundenschnell durchsuchen und auf dem Bildschirm anzeigen. Mehrere Datensätze mit gleichem Suchwort werden alphabetisch aufgelistet.

Damit ist die Leistungsfähigkeit keineswegs ausgeschöpft. Man kann auch verschiedene Suchbegriffe miteinander verknüpfen und so zum Beispiel bei einer Adreßdatei alle Leute, die in München wohnen und Meyer heißen, aussuchen. Man nennt dieses auch eine UND-Verknüpfung. Aber auch ODER-Verknüpfungen sind möglich. Beispielsweise lassen sich so



Bild 6. Ein Datensatz der StarDatei auf dem Bildschirm

alle Leute, die in München oder in Augsburg wohnen, bei einer angelegten Adreßdatei herausfinden. Die UND- beziehungsweise ODER-Verknüpfung kann im Suchbegriff auch mehrmals vorkommen, aber leider nicht gemischt.

Es können natürlich nicht nur die Titelzeilen durchsucht werden, sondern auch einzelne Teile aus den Datensätzen selber. Das dauert aber länger, da dazu auf die Diskette zugegriffen werden muß. Jeder gefundene Datensatz wird dann wieder in alphabetischer Reihenfolge auf dem Bildschirm aufgelistet und man kann auswählen, ob man ihn in den Speicher übernehmen will. Dabei lassen sich auch einzelne Datensätze markieren, beispielsweise um sie dann später auszudrucken oder mit »StarTexter« weiterverarbeiten zu lassen.

Bemerkenswert gut gelöst sind die diversen Möglichkeiten, sich einen Überblick der vohandenen Daten zu verschaffen.

Die möglichen Fähigkeiten, die die einzelnen Drucker anbieten, werden bei weitem nicht voll ausgeschöpft. Das sind zum Beispiel Unterstreichen, Fettschrift oder auch das Format des Ausdruckes, das sich nicht frei wählen läßt.

Sinnvoll dagegen ist die Möglichkeit, sich jederzeit eine Hardcopy des Bildschirms ausgeben lassen zu können.

Nachteilig ist, daß sich aus dem Inhaltsverzeichnis einer Datendiskette nicht erkennen läßt, wieviel Speicherkapazität noch zur Verfügung steht. Man muß sich bei Bedarf einen Statusreport geben lassen, der den prozentualen Speicherverbrauch angibt. Ferner wird die Anzahl der im Moment gespeicherten Datensätze angezeigt.

Wichtig bei Dateiverwaltungsprogrammen ist vor allem der Datenexport, also die Weitergabe gesammelter Daten an andere Programme, zum Beispiel Textverarbeitungen. Leider arbeitet »StarDatei« nur mit dem »StarTexter« aus gleichem Hause zusammen. Der »StarTexter« kann die in dem speziellen Format abgelegten Daten lesen und in seinem Textspeicher übernehmen oder auch beim Ausdrucken von Formbriefen verwenden.

Gut gelöst ist die Möglichkeit, einzelne Datenfelder mittels spezieller Zeichen für »StarTexter« zu definieren. Die so festgelegten Datenfelder lassen sich in beliebiger Reihenfolge von »StarTexter« weiterverarbeiten, so daß über die Textverarbeitung Listendruck von Datensätzen oder diverse Serienbriefe nach unterschiedlichsten Verarbeitungskriterien möglich ist. Die Übernahme von Daten durch »StarDatei« in seinen Speicher ist nur eingeschränkt möglich. Man kann eine SEQ-Datei in den Arbeitsspeicher einlesen und auf dem Bildschirm anzeigen lassen. Diese Daten lassen sich dann als Datensatz speichern, wenn sie nicht mehr als drei Blöcke auf der Diskette belegen.

Man kann jederzeit eine mathematische Berechnung durchführen. Dazu gibt man die Berechnung ins Datenfeld ein

und drückt CTRL und »=«. Sofort bekommt man das Ergebnis in der nächsten Zeile darunter angezeigt. Es kann im Datensatz weiterverarbeitet werden. Selbst kleine Basic-Anweisungen innerhalb der »StarDatei« sind kein Problem. Wem das allerdings immer noch zu wenig ist, der kann für kurze Zeit »StarDatei« ganz verlassen und ein wenig Basic programmieren. Man muß allerdings darauf achten, daß man nach jeder Kommandozeile ein »:STOP« anfügt, ansonsten wird »StarDatei« sofort wieder aktiviert.

Die oberste Bildschirmzeile ist die Kommunikationszeile, wenn man Anweisungen im Text übernehmen möchte. Eine hier plazierte Berechnung wird von »StarDatei« übernommen.

Jeder Ausdruck auf einem Drucker setzt die richtige Anpassung an dessen Fähigkeiten und Eigenheiten voraus. Da »StarDatei«-Druckmöglichkeiten wie Fettdruck oder Unterstreichung nicht unterstützt, fällt diese Druckeranpassung mager aus. Das Programm fragt dabei im Menü »Parameter« nach den wichtigsten Druckereinstellungen wie Wandlungsart, automatischer Zeilenvorschub, Anzahl der Spalten zum Einrücken beim Ausdruck und noch einiges mehr. Man kann hier wahlweise Drucker über den seriellen Bus oder über eine am User-Port simulierte Centronics-Schnittstelle ansprechen. Die Druckcodes von Umlauten, die man im Text frei verwenden kann und die normalerweise auf den Funktionstasten liegen, lassen sich hier ebenfalls festlegen. Des weiteren ist der von jeder Textverarbeitung her bekannte Zeilenumbruch, der beim Schreiben ein zu lang geratenes Wort vollständig in die nächste Zeile zieht, möglich. Diesen Zeilenumbruch kann man wahlweise ein- oder ausschalten.

Normalerweise entspricht die Tastaturbelegung des C64 nicht der deutschen Schreibmaschinen-Norm (QWERTZ). Um hier jedoch trotzdem alle Umlaute eingeben zu können, wurden sie auf die Funktionstasten gelegt. Wem das zu umständlich ist, kann auch auf eine deutsche Tastenbelegung umschalten, wobei jetzt alle Zeichen dort liegen, wo sie auch bei einer deutschen Schreibmaschine zu finden sind. Auch »z« und »y« sind vertauscht.

Selbstverständlich lassen sich die Bildschirmfarben an individuelle Neigungen anpassen. Interessant und auch von »StarTexter« her bekannt ist die Möglichkeit, einen beliebigen Zeichensatz auszuwählen, der dann beim nächsten Neustart des Programms mit eingeladen wird. Diese ganzen Parametereinstellungen lassen sich speichern und werden automatisch wieder aktiviert, wenn das Programm neu eingeladen und gestartet wird.

Für einfachere Datenverwaltung, insbesondere das Führen von Adreßdateien oder auch beispielsweise Plattensammlungen, läßt sich »StarDatei« sehr gut verwenden. Wer zudem schon das Textverarbeitungsprogramm »StarTexter« besitzt, kann die gesammelten Daten sehr gut übernehmen und mannigfaltig weiterverarbeiten. Dazu kommt aber das Problem, Daten nicht frei an andere Textverarbeitungsprogramme weitergeben zu können. Auf der anderen Seite überzeugen jedoch Leistungen wie das einfache Eingeben neuer Datensätze ohne umständliche Maskendefinition mit endgültiger Festlegung aller Datenfelder. Man bekommt jedesmal einen Textbereich zur freien Verfügung gestellt und kann jederzeit die Organisation eines Datensatzes umstellen, sogar eine Notizzettelsammlung mit ungeordneten Datensätzen aufstellen.

Ferner bemerkenswert ist die Möglichkeit, kurzzeitig das Programm zu verlassen und komplizierte, mathematische Berechnungen schnell in Basic durchzuführen. Bei Betrachtung all dieser Fähigkeiten bekommt man für 64 Mark ein gutes Programm.

Welches Programm für Sie das richtige sein wird, hängt stark vom Anwendungsgebiet und den finanziellen Möglichkeiten ab. Als guter Tip sei aber gesagt, daß Sie das Programm ruhig »eine Nummer größer« kaufen können, da oftmals mit zunehmender Erfahrung der Anspruch an das Programm wächst.

(Guido Weckwerth/bj/cg)

Verwendungszweck	Name	Preis	Bezugsquelle
Textverarbeitung C64	Vizawrite	Disk: 238,– Modul: 298,–	Data Technology Management Data Technology Management
	Textomat Plus	99,–	Data Becker
	Star Texter	64,–	Sybex-Verlag GmbH
Textverarbeitung C128	Wordstar	199,–	Markt&Technik
	Superscript	198,–	Commodore
	Protext	89,–	Markt&Technik
	Textomat Plus	99,–	Data Becker
Dateiverwaltung C64	Superbase 64	198,–	Data Becker
	Star Datei	64,–	Sybex-Verlag GmbH
Dateiverwaltung C128	Superbase 128	198,–	Commodore
	dBase II	199,–	Markt&Technik
Kalkulation C128	Multiplan 128	199,–	Markt&Technik

Info:

Commodore — Lyoner Str. 38, 6000 Frankfurt/Main 71, Tel. (069) 6638-0
Data Becker — Merowinger Str. 30, 4000 Düsseldorf, Tel. (0211) 310010
Data Technology Management — Bornhofenweg 5, 6200 Wiesbaden, Tel. (06121) 407989
Markt&Technik — Hans-Pinsel-Str. 2, 8013 Haar bei München, Tel. (089) 4613-205
Sybex-Verlag GmbH — Postfach 300961, 4000 Düsseldorf 30, Tel. (0211) 61802-13



Befehlserweite

die

Das Basic V2.0 des Commodore ist etwas schmächtig geraten. Mit den hier vorgestellten Befehlserweiterungen wird Ihr C64 zur wahren Wundermaschine.

Jedem Anwender, der schon versucht hat, intensiver mit dem C64 zu arbeiten, wird das unterdimensionierte Basic V2.0 des Commodore zu klein geworden sein. Es fehlen einfach die nötigen Befehle, um die Grafik, die Musik und einige andere wichtige Routinen des Computers ohne viel Aufwand anzusprechen. Ein einfacher Befehl wie etwa CIRCLE zum Zeichnen eines Kreises ist leicht zu verstehen und ersetzt viele unnötige und schwierige eigene Programmteile.

Für diesen Zweck wurden spezielle Programme geschrieben, die den Basic-Befehlssatz erweitern und damit dem Anwender zur Verfügung stehen. Manche Basic-Erweiterungen unterstützen das Programmieren, andere stellen spezielle Grafik-Befehle zur Verfügung und einige besitzen sogar beides.

Hier erhalten Sie eine Übersicht über die gebräuchlichsten Basic-Erweiterungen, die Ihnen helfen werden, Ihren Computer besser auszunutzen.

GBasic

Die Erweiterung GBasic wird als Modul geliefert, so daß sämtliche Befehle, die GBasic zu bieten hat, sofort nach Einschalten des Computers zur Verfügung stehen. Das Modul belegt 8 KByte des Speichers, so daß nur noch 30719 Bytes freier Basic-Speicher übrig bleiben.

An Grafikbefehlen stellt GBasic die üblichen Befehle wie PLOT, LINE, BOX, FILL, CIRCLE, etc. zur Verfügung. Mit diesem Befehlssatz ist man ganz gut bedient. Die Ausführung der Befehle erfolgt relativ schnell, nur bei den Kreisfunktionen fällt die Geschwindigkeit etwas ab.

Einen interessanten Befehl besitzt GBasic, der unseres Wissens in keiner anderen Grafikerweiterung zu finden ist. Dies ist der Befehl VEKTOR. Er zieht eine Linie, die durch die Anfangs- und Endkoordinaten festgelegt ist. Anders als beim LINE-Befehl stoppt VEKTOR jedoch mit dem Ziehen der Linie, sobald er auf einen Punkt mit der gleichen Farbe trifft.

Neben den Befehlen für hochauflösende Grafik besitzt GBasic weiterhin Befehle zum Editieren und Steuern von Sprites. GBasic unterstützt durch mehrere Befehle auch das Arbeiten mit Lichtgriffel oder Grafiktablett. Mit GBasic bekommt man eine Menge für sein Geld geliefert. Das GBasic-Modul kann man als Normal- oder Turbo-Modul erhalten. Die Bezeichnung Turbo bezieht sich auf den eingebauten Turbolader für die Floppy.

Simons Basic

Die wohl älteste und bekannteste Befehlserweiterung dürfte Simons Basic sein. Simons Basic ist auf Modul erhältlich und zählt zu den Erweiterungen, die Funktionen für jedes Anwendungsgebiet bereitstellen (über 100 Befehle jeder Art).

Simons Basic kann einen HiRes-Schirm verwalten und zwar sowohl im Normal- wie auch im Multicolor-Modus. Befehle zur Unterstützung von Lichtgriffel und Grafiktablett sind ebenfalls integriert.

In der Geschwindigkeit tut sich Simons Basic nicht besonders hervor. Es zählt eher zu den langsamen Programmen. Positiv dürfte wohl die weite Verbreitung sein. Deshalb sind in den verschiedenen Computerzeitschriften auch gelegentlich Listings mit dieser Erweiterung abgedruckt. Das Handbuch ist mit über 70 Seiten zur Einführung recht gut geeignet. Es wird aber nur jeweils die genaue Syntax der Befehle erklärt. Für einen erfahrenen Programmierer ist dies kein Handicap. Ein Anfänger jedoch sollte ein entsprechendes Handbuch zu Rate ziehen. Gerade für Simons Basic sind jede Menge gute Bücher erhältlich.

Macro-Basic

Ein völlig anderer Weg, das Commodore-Basic um neue Befehle zu bereichern, wird von Macro-Basic beschritten. Aus einer Sammlung kleiner Erweiterungsmodule läßt sich jede mögliche Kombination von Befehlen zusammenstellen. Man kann also die nur für ein Programm nötigen Befehle zu einer Befehlserweiterung zusammenfassen. Es sind etwa 160 Befehle verfügbar. Alle Befehlsmodule (außer dem Grafikmodul) sind kompatibel zu den anderen Commodore-Computern. Jedes Macromodul entspricht einem einzelnen Befehl, nur das Grafikmodul fällt etwas aus der Rolle. Im Grafikmodul sind alle 23 Grafikbefehle zu einem großen Modul zusammengefaßt. Da beim Programmieren von Grafiken in der Regel sowieso alle Befehle gebraucht werden, ist dies nicht weiter tragisch. Außerdem ist das Grafikmodul nur auf dem C64 lauffähig.

Die Dokumentation ist auf Grund der vielen Befehle sehr umfangreich. Generell wird pro Befehl etwa eine DIN-A4-Seite zur Erklärung gebraucht. Den Grafikbefehlen wird weniger Platz gewidmet. Auf einer Seite werden zwei bis drei Befehle erläutert. Trotzdem reicht auch hier noch der Platz für kurze Beispielprogramme.

rungen, jedem helfen

Hypra-Basic

Auf genau der gleichen Welle liegt auch Hypra-Basic (64'er, Heft 4/86). Hier können ebenfalls aus einzelnen Modulen Basicerweiterungen für jeden Zweck geschaffen werden. Man stellt sich wie bei einem kalten Buffet seine eigene, auf eine ganz bestimmte Aufgabe zugeschnittene Erweiterung zusammen, die keine überflüssigen Befehle mehr enthält.

Die Funktionen belegen keinen Basic-Speicherplatz, so daß die vollen 38911 Bytes verfügbar sind.

Durch die universelle Einsetzbarkeit der Funktionen und Routinen können damit alle Anwendungsbereiche abgedeckt werden.

Grafik 2000

Dies ist eine reine Grafikerweiterung, die 41 neue Befehle umfaßt. Grafik 2000 verwaltet zwei Grafikschirme. Die Erweiterung selbst und die Grafikschirme liegen außerhalb des Basic-Speichers, so daß dem Anwender die vollen 38911 Byte zur Verfügung stehen.

Alle Befehle lassen sich wie beim Basic V2.0 abkürzen, da sie als Token gespeichert werden; selbst nach einem THEN ist kein Doppelpunkt nötig. Als besonderer Befehl kann bei Grafik 2000 der Befehl WINDOW genannt werden. Er ermöglicht die Darstellung eines Grafikfensters im Textmodus. Dies ist sehr nützlich, wenn man zum Beispiel ein Adventure programmieren möchte. Rahmen- und Hintergrundfarben können ohne Unterbrechung des laufenden Programms durch einen Druck auf die Restore-Taste gesetzt werden. Grafik 2000 wurde im Grafik/Drucker-Sonderheft 4/85 als Listing zum Abtippen abgedruckt.

Datawork Basic

Diese Erweiterung unterstützt den Programmierer von Dateiverwaltungen oder Textverarbeitungen. Die 22 enthaltenen Befehle zielen hauptsächlich auf Stringbehandlung und Datenausgabe-Erleichterung. Die Routinen liegen in von Basic aus nicht erreichbaren Speicherbereichen, so daß kein Basic-Speicher verloren geht.

Mit der Erweiterung lassen sich zusätzlich komfortabel Bildschirmmasken erstellen und einfach auf den Schirm bringen. Ebenfalls integriert ist eine Centronics-Schnittstelle, die es ermöglicht, Nicht-Commodore-Drucker ohne Interface an den C64 anzuschließen.

Noch ein Vorteil: Mit den Cursor-Tasten kann ein Listing hin- und hergeschoben werden. Datawork-Basic finden Sie als Anwendung des Monats im 64'er, Ausgabe 1/86.

MB-Highway

Das Basic V7.0 des Commodore 128 ist zwar sehr reichhaltig, doch ist noch nicht alles enthalten. Abhilfe schafft MB-Highway für den C128-Modus. Es reizt Ihren C128 mit etwa 220 (in Worten: zweihundertzwanzig!) zusätzlichen Befehlen wohl bis in die Knochen aus. Es sind so viele Funktionen aller Gattungen vertreten, daß man schon eine ganze Weile braucht, bis man alle ausprobiert und Einsatzgebiete gefunden hat. Mit dieser Erweiterung dürfte der C128 in Sachen Basic wohl nicht mehr zu schlagen sein.

Info: GBasic - Omikron, Erlachstr. 15, 7534 Birkenfeld. Modul+Diskette, 259 Mark.
Simons Basic – Commodore, Lyoner Str. 38, 6000 Frankfurt 71. Modul, 59 Mark.
Grafik 2000, Datawork-Basic, Hypra-Basic – Markt&Technik Verlag, Hans-Pinsel-Str. 2a, 8013 Haar. Programm-Service-Disketten, jeweils 29,90 Mark.
MB-Highway – SAS H.J. Bernd, Langgasee 93, 5216 Niederkassel 5. Modul, 298 Mark

Was Sie später brauchen könnten

Tools und *Utilities* sind die Werkzeuge der Programmierer. Ohne diese Hilfsprogramme wird ein effektives Programmieren fast unmöglich. Lesen Sie hier, was Sie später einmal benötigen könnten.

Für das Programmieren mit dem Computer sind die *Tools* eine große Hilfe. Darunter sind Befehls-Erweiterungen zu verstehen, die einem das Arbeiten und Programmieren stark vereinfachen. Eingedeutscht könnte man sie auch Programmierhilfen nennen. In ihnen finden sich Routinen wie zum Beispiel AUTO (Zeilennummerierung), FIND (um bestimmte Zeichen in einem Programmtext zu finden), DELETE (löscht ganze Zeilenblöcke), TRACE (überwacht und dokumentiert das gerade ablaufende Programm), DUMP (zeigt alle zu diesem Zeitpunkt definierten Variablen an) und RENUM (Umnummerieren eines Programmes). Die Liste ließe sich fast endlos fortführen, doch sollen Sie hier ja nur eine ungefähre Vorstellung erhalten.

Aber mit Basic ist aus diesem Computer trotz alledem nicht alles herauszuholen. Die besten Leistungen lassen sich eben nur dann erreichen, wenn man sich den Umweg über den Basic-Interpreter erspart und gleich direkt so programmiert, wie es der Computer am besten versteht, also in Assembler (oft auch Maschinensprache genannt). Maschinenspracheprogrammierer können auf *Monitore* (dies sind keine Fernsehschirme!) und *Assembler* nicht mehr verzichten. Diese Hilfsprogramme gewähren dem Anwender Einblicke in alle Speicherbereiche, die er damit auch ändern kann. Er erhält über diese *Utilities* Informationen, wie etwa verschiedene Routinen aufgebaut sind. Ebenfalls sind diese Programme unerläßlich, wenn es darum geht, selber in Maschinensprache zu programmieren. (dm)

Grafikprogramme

Die Farbgrafikeigenschaften des C64 sind wohl mit ein Grund für seinen herausragenden Erfolg. Leider sind sie nur schwer zu programmieren. Wir stellen Ihnen hier einige Zeichenprogramme vor, die es auch dem Ungeübten leicht machen, die schönsten Grafiken einfach zu erstellen.

Egal, ob Sie nur zu Ihrem Vergnügen etwas am Bildschirm zeichnen wollen oder Bilder benötigen, um sie in eigene Programme einzubinden: Ein Zeichenprogramm ist auf alle Fälle eine lohnende Investition.

Wohl in jedem besseren Zeichenprogramm sind folgende Funktionen enthalten:
- Kreise ziehen (CIRCLE)
- Flächen ausfüllen (PAINT, FILL)
- Freihand-Zeichnen (DRAW)
- Punkte verbinden (LINE, RAYS)
- ausschnittweise Vergrößerung (ZOOM)
- Rechtecke zeichnen (RECTANGLE)
- Punkte und Linien löschen (ERASE)
- Bilder laden und speichern (LOAD, SAVE)
- Bildteile kopieren, spiegeln und verschieben (MOVE, MIRROR, COPY)
- ausgefüllte Rechtecke zeichnen (BOX)
- mindestens zwei Grafik-Bildschirme, zwischen denen man umschalten kann (SWAP)
- die Steuerung des Grafik-Cursors erfolgt entweder mit dem Joystick, den Cursortasten, einem Lightpen, über die Maus oder mit einem Stift auf einem Grafiktablett.

Von der Leichtigkeit der Bedienung hängt auch die Qualität des verwendeten Grafikprogrammes ab. So dürfte wohl ein Zeichen-Cursor, der langsam, aber stetig, beschleunigt (das bedeutet: Kurze Wegstrecken werden langsamer überwunden, so daß man sich im Bild auch punktweise bewegen kann), besser geeignet sein als ein »High-Speed-Cursor«. Gleichwohl positiv zu werten ist auch eine Menü-Seite, über die die einzelnen Funktionen und Modi komfortabel ausgewählt werden können.

Diese Übersicht der empfehlenswertesten Zeichenprogramme soll Ihnen helfen, anhand der Stärken und Schwächen der vorgestellten Produkte das für Sie geeignete Grafikprogramm herauszufinden.

Hi-Eddi

Hi-Eddi ist ein Programm, das konzipiert wurde, um mit möglichst wenig Aufwand Grafiken aller Art erstellen zu können. Neben den schon oben beschriebenen Funktionen steht noch ein komfortabler Sprite-Editor zur Verfügung. Das Besondere an ihm ist wohl die Möglichkeit, entwickelte Sprites ganz einfach in den Zeichen-Bildschirm hineinzusetzen, zu vervielfältigen, zu löschen und in alle Richtungen zu drehen oder zu spiegeln. Insgesamt sieben verschiedene Grafik-Bildschirme stehen dem Anwender zur freien Verfügung. Dabei ist es unter anderem problemlos möglich, Teile aus einem Bild in ein anderes zu übertragen.

Eine andere, reizvolle Funktion besteht in der Möglichkeit der Animation. Das bedeutet, daß mehrere, in Einzelphasen verschiedene Bilder der Reihe nach an der gleichen Bildschirmposition eingeblendet werden, so daß der Eindruck einer Bewegung entsteht. Dadurch, daß sich auch Schrift in die Bilder einsetzen läßt, eignet sich Hi-Eddi sehr gut dazu, zum Beispiel Schaltpläne oder Grundrisse von Gebäuden zu erstellen. Hi-Eddi war Listing des Monats im 64'er, Ausgabe 1/85 und ist wohl das beste und beliebteste Grafikprogramm, das je in einer Zeitschrift veröffentlicht wurde. Für die meisten Drucker wurden im Laufe der Zeit Hardcopy-Routinen entwickelt und nach und nach im 64'er veröffentlicht.

Im Laufe der Zeit erfährt jedes gute Anwender-Programm Verbesserungen. So auch Hi-Eddi. Die neue Version »Hi-Eddi+« wurde um die folgenden Funktionen erweitert: JOTS (in etwa die Funktion einer Sprühdose), ZOOM und ein Drehen der Text-Schreibrichtung. Ein weiteres Plus ist die in beiden Versionen vorhandene Möglichkeit, sich seine individuellen Menütafeln zu erstellen und die gezeichneten Bilder auch auf verschiedenen Druckern auszugeben.

Koala Painter

Der Oldtimer unter den Grafikprogrammen dürfte der Koala Painter sein. Er war eines der ersten Zeichenprogramme, die für den Commodore 64 angeboten wurden. Natürlich beherrscht der Koala Painter alle im Vorspann genannten Funktionen, auch wenn die Kreise nie wirklich kreisförmig werden. Als Zubehör bekommt man das Koala Pad angeboten, eine Art Tablett, mit dem es auf einfachste Art möglich ist, Zeichnungen vom Papier auf den Bildschirm zu bringen. Es muß nur das Blatt Papier auf die Zeichenfläche des Pad gelegt und die Linien mit dem Pad-Griffel nachgezogen werden. Synchron zu den Bewegungen überträgt das Pad die Formen in den Computerspeicher und damit auch auf den Bildschirm. Durch die einfache Handhabung und das komfortable Hauptmenü wird der Koala Painter zum bedienungsfreundlichen Bildeditor. Als Bonbon bietet die Herstellerfirma den Koala Printer an, der es ermöglicht, die mit dem Koala Painter gezeichneten Bilder auf fast jedem Drucker auszudrucken.

Profi Painter

Dieses Konstruktionsprogramm ist grundsätzlich ähnlich wie Hi-Eddi/Hi-Eddi+ entworfen worden. Der Aufbau der (immer zu sehenden) Menüleiste jedoch ist auf dem neuesten Stand der Programmierung. Man muß nur mit dem Grafik-Cursor das betreffende Symbol »anclicken«, und schon öffnet sich ein Bildschirmausschnitt (»Window«), in dem man die gewünschte Funktion anwählen kann. Das Programm arbeitet leider nicht farbig, doch können Flächen in verschiedenen Schraffierungen ausgefüllt werden. Die gute Bedienung macht den Profi Painter zu einem optimalen Grafik-Werkzeug.

Star Painter

Fast genauso wie der Profi Painter ist auch der Star Painter aufgebaut. Er verfügt ebenfalls über eine Benutzerführung, die bequem und einfach zu bedienen ist. Der augenfälligste Unterschied zu den anderen Programmen ist wohl der, daß hier nur ein Grafik-Bildschirm existiert. Dafür umfaßt dieser jedoch 640 x 344 Bildpunkte, wovon aber nur 256 x 168 Bildpunkte auf einmal im Anzeigefenster zu sehen sind. Der Rest wird aus dem aktuellen Bildschirm »hinausgeschoben«. Kreise werden sehr präzise und rund gezeichnet. Wie auch

Hi-Eddi/Hi-Eddi+ stellt der Star Painter einen kleinen Sprite-Editor zur Verfügung. Die hiermit entworfenen Sprites lassen sich in das zu entwickelnde Bild einmontieren, wie auch Bestandteile des Bildes in den Sprite-Editor übernommen und verändert werden können. Der Star Painter ist ebenfalls nicht fähig, mehrere Farben darzustellen. Es stehen aber 18 verschiedene Schraffierungen zur Verfügung, mit denen sich Flächen ausfüllen lassen.

Blazing Paddles

Blazing Paddles gehört ebenfalls zu den komfortablen Mehrfarb-Zeichenprogrammen. Es besitzt neben den Standardfunktionen noch folgende Optionen: SPRAY (simuliert eine Spraydose), SHAPES (damit lassen sich Formen festlegen und in das Bild einfügen), TEXT (um Zeichen einzusetzen), WINDOW (damit wird es möglich, auf der Diskette bestehende Formen in das Bild zu laden) sowie eine komfortable und von der Auswahl der Druckertypen reichhaltige Druckerroutine, die selbständig die Farbwerte in abdruckbare Graustufen umrechnet. Als Eingabegeräte lassen sich die Tastatur, ein Lightpen, Trackball, Joystick, Paddles, eine Maus oder ein Grafiktablett verwenden. Alles in allem ein sehr gutes Werkzeug zum Erstellen farbiger Grafiken.

Paint Magic

Dieses Programm ist zwar schon einige Zeit auf dem Markt, doch hat es noch nichts von seiner Anziehungskraft verloren. Das Hauptmenü macht vielleicht keinen so professionellen Eindruck wie die anderen vorgestellten Programme, doch läßt sich mit ihm bequem arbeiten. Der Befehlssatz besteht nur aus den im Vorspann erwähnten Funktionen, wobei einige Befehle, die mit den Farben zu tun haben, erst nach einer kleinen Einarbeitungszeit deutlich werden. Trotz der im Vergleich zu modernen Zeichenprogrammen nicht überragenden Auswahl an Befehlen und Funktionen können die Bilder doch mit dem gewohnten Standard mithalten.

Info: **Hi-Eddi** – Markt&Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar (64'er, Heft 1/85; 64'er Sonderheft 6/85 – Programm-Service-Diskette Heft 1/85, 29,90 Mark).
Hi-Eddi+ – Markt&Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar. Buch mit Diskette, 48 Mark.
Koala Painter – Harman Deutschland, Hünderstr. 1, 7100 Heilbronn, Koala Pad, Grafik-Tablett mit Diskette/Kassette, 240 Mark, Koala Printer, Diskette, 95 Mark.
Profi-Painter – Data Becker, Merowingerstr. 30, 4000 Düsseldorf. Diskette, 99 Mark
Star Painter – Sybex-Verlag, Vogelsanger Weg 111, 4000 Düsseldorf 30. Diskette, 64 Mark.
Blazing Paddles – Softville, Schwarzwaldstr. 8a, 7602 Oberkirch. Diskette, 139 Mark.
Paint Magic – Markt&Technik Verlag AG, Hans-Pinsel-Str. 2, 8013 Haar. Diskette, 59 Mark.

Einige Begriffserklärungen zum Text:

Unter die Gattung der Eingabegeräte fällt das **Grafiktablett.** Es besteht aus einer Platte, in der Drähte in Form eines Fliegengitters eingegossen sind, sowie einem Zeichenstift. Werden nun mit dem Zeichenstift Linien, die zum Beispiel von einem aufgelegten Blatt Papier stammen könnten, nachgefahren, so wird die Bewegung in den Computer übertragen.

Der Begriff **HiRes** ist eine Abkürzung für die High Resolution, auf Deutsch : hohe Auflösung. Das Pendant daszu ist LoRes für Low Resolution = niederige Auflösung. Im LoRes-Modus kann man nur die Zeichen eingeben, die auch auf der Tastatur zu sehen sind, und er ist der Modus, in dem sich der C 64 nach dem Einschalten befindet. Dort ist aber kein punktweises Zeichnen möglich. Um aber punktweise Zeichnen zu können, ist es nötig, in einen anderen, eben diesen **HiRes**-Modus, zu gehen. Dort offenbart sich einem dann auch der **Grafik-Bildschirm**, der nicht mehr wie vom Textbildschirm her gewohnt 1000 Zeichen, sondern 64000 einzelne Punkte darzustellen vermag.

Wem der Begriff **Grafik-Cursor** ebenfalls nichts sagt, der stelle sich darunter ein Sprite in Kreuz- oder Pfeilform vor, dessen Mitte oder Spitze auf den Punkt deutet, der gezeichnet werden soll. (dm)



Ein Lied kommt aus dem Chip

Wenn Sie den Computer zur Musikmaschine machen wollen, dann brauchen Sie nur das richtige Programm. Wir stellen Ihnen die vier bekanntesten Musik-Programme für den C64 vor.

Der C64 ist mit einem der besten Sound-Chips ausgestattet, die es auf dem Computer-Markt gibt. Die geradezu fantastische Musikbegleitung bei manchen professionellen Spielen beweist dies immer wieder. Nun möchten Sie sicherlich auch gerne die musikalischen Fähigkeiten des C64 ausnutzen. Aber die Programmierung von Geräuschen oder ganzen Musikstücken ist in Basic sehr schwer, wenn nicht gar unmöglich. Abhilfe schafft ein Musikprogramm, mit dem man Musik eingeben und spielen kann, ohne von Computern und deren Programmierung eine Ahnung haben zu müssen. Vier von diesen Programmen wollen wir Ihnen kurz vorstellen. Wenn Sie sich für diese Thematik interessieren, lassen Sie sich die Programme doch mal bei Ihrem Fachhändler vorführen.

Eines der ältesten, aber trotzdem besten Programme ist das »Music Construction Set« von Electronic Arts. Dieses Programm verhält sich, wie alle anderen auch, wie eine Textverarbeitung für Musik. Auf einem am Bildschirm angezeigten Notenblatt können Sie Noten setzen und löschen, umstellen, kopieren und dann natürlich auch spielen lassen. Für den guten Ton stehen beim »Music Construction Set« 13 verschiedene Klangfarben zur Verfügung. Das Programm ist sehr einfach zu bedienen, hat eine deutsche Bedienungsanleitung und ist voll Joystick-gesteuert.

Ein weiteres, Joystick-gesteuertes Programm ist der »Music Shop« von Broderbund. Es bietet mehr Möglichkeiten als das »M.C.S.«, da man die Klangfarben selber programmieren kann. Außerdem bietet »Music Shop« zur Benutzerführung Windows (Bildschirmfenster), was den Umgang besonders einfach macht. Besitzer eines grafikfähigen Druckers können die Notenblätter ausdrucken und sich so ihre eigenen Liederbücher herstellen.

Auch von der Firma Activision gibt es ein Musikprogramm. Es heißt »Music Studio« und zeichnet sich durch besonders bunte Bildschirmmenüs aus. Außerdem kann es auch von Leuten benutzt werden, die sich nicht mit dem Notensystem auskennen, denn man kann Musik auf dem Bildschirm »malen«. Auch hier kann man Lieder drucken und vorher sogar mit Texten versehen. Die Benutzerführung erfolgt mit vielen bunten Menüs und Joystick.

Das neueste und umfangreichste Programm heißt »The Music System« und kommt von der englischen Firma Firebird. Es bietet von allen vorgestellten Programmen am meisten und ist trotzdem sehr einfach zu bedienen. Die über achtzig Seiten des deutschsprachigen Handbuchs tun einiges Gutes dazu. »The Music System« ist in mehrere Module unterteilt. Dort kann man Musik per Notenblatt eingeben oder aber wie auf einer Orgel spielen und aufnehmen. Mehrere Musikstücke können aneinander gebunden werden. Außerdem kann das »Music System« mit einem angeschlossenen Synthesizer zusammenarbeiten, wenn man ein sogenanntes MIDI-Interface besitzt.

Ein MIDI-Interface dient zur Kopplung von professionellen Synthesizern an den C64. Der Computer spielt dabei selber keine Musik, er steuert nur die anderen angeschlossenen Geräte. Der Klang eines Synthesizers ist um Klassen besser als der des SID. Allerdings darf man für dieses Klangerlebnis auch nicht geizig sein: Ein komplettes MIDI-System kostet zwischen ein- und zehntausend Mark.

Für diese MIDI-Interfaces gibt es dann auch von den Herstellern die entsprechende Software, die aber meistens für Musiker entwickelt wurde und für Laien sehr unübersichtlich ist. Eine löbliche Ausnahme ist das Programm »Track Star« von Steinberg Research, das gerade für MIDI- und Musikanfänger geschrieben wurde.

Alle vier Musikprogramme haben ihre Vor- und Nachteile, dazu gehört auch der Preis. Denn gerade bei der Musik-Software gilt, daß man Leistung auch noch relativ teuer bezahlen muß. Dafür erhält man aber auch Programme, mit denen man Musik machen kann, und nicht nur bloßes Rum-Gedüdel.

(bs/tr)

Info:
Music Construction Set, Music Shop, Music Studio: AriolaSoft, Postfach 1350, 4830 Gütersloh 1
The Music System: Rushware, An der Gümpgesbrücke 24, 4044 Kaarst 2
Track Star: TSI, Neustraße 12, 5481 Waldorf

Checksummer 64 V3

Der Checksummer 64 V3 überprüft jede Basic-Zeile direkt nach der Eingabe, erkennt Fehleingaben sowie Vertauschungen von Ziffern und erspart eine aufwendige Fehlersuche.

Der Checksummer 64 V3 ist ein kleines Maschinenprogramm, das Sie sofort unterrichtet, ob Sie die jeweilige Programmzeile korrekt eingegeben haben.

So gehen Sie vor:

1. Programm abtippen und speichern.

2. Starten mit RUN.

3. Nach kurzer Zeit sehen Sie am Bildschirm: CHECKSUMMER 64, CHECKSUMMER AKTIVIERT, AUSSCHALTEN MIT POKE 1,55, ANSCHALTEN MIT POKE 1,53, READY.

4. Anschalten des Checksummer 64 V3 mit POKE 1,53.

5. Test: Geben Sie in einer freien Zeile ein: »1 REM« und drücken die RETURN-Taste. Am Bildschirm oben links sollten Sie die Prüfsumme <63> sehen.

6. Geben Sie ein Listing aus unserem Heft ein. Nach jeder Zeile wird die Zahl, die im Listing in Klammern < > steht, in den Bildschirm eingeblendet. Stimmen die Zahlen nicht überein, so liegt vermutlich ein Eingabefehler vor. **Die Zahl in den Klammern, und auch die Klammern selbst, dürfen beim Abtippen nicht mit eingegeben werden!**

7. Der Checksummer 64 V3 bemerkt auch Vertauschungen von Zahlen und Buchstaben, aber nicht das Fehlen (oder Hinzufügen) von Leerzeichen.

8. Unsere Basic-Listings enthalten keine Steuerzeichen mehr. Diese werden ersetzt durch Klartext und stehen zwischen geschweiften Klammern. Deshalb sind weder die Klammern noch was dazwischen steht, abzutippen, sondern die in Tabelle 1 aufgeführten Tasten zu drücken. Auf Ihrem Bildschirm erhalten Sie dann wieder die entsprechenden Grafikzeichen.

9. Alle Grafikzeichen werden ebenfalls ersetzt durch unterstrichene oder überstrichene Großbuchstaben.

Unterstrichene Buchstaben bedeuten, daß Sie die SHIFT-Taste und den angegebenen Buchstaben drücken müssen, überstrichene jedoch die Commodore-Taste mit dem Buchstaben. Auch hier erhalten Sie am Bildschirm das entsprechende Grafikzeichen und nicht etwa das im Listing erkennbare Zeichen.

Die Leerzeichen zwischen den einzelnen Basic-Befehlen können beim Abtippen entfallen (ohne Einfluß auf die Checksumme zu nehmen). Dies ist besonders bei speicherkritischen Programmen wichtig. Ebenso müssen Zeilen, die mehr als 80 Zeichen pro Zeile enthalten, mit den bekannten Abkürzungen für die Basic-Befehle (siehe auch das Handbuch zum C64, Anhang D, Seite 130) eingegeben werden.

Sie können die Programme auch weiterhin ohne den Checksummer eintippen. (F. Lonczewski/gk)

Hinweis: {13 SPACE} bedeutet 13mal die Leertaste drücken

```
9 REM *********************************
10 PRINT"{CLR,11SPACE,RVSON}CHECKSUMMER 64
   V3{RVOFF}"
11 PRINT"{2DOWN,9SPACE}EINEN MOMENT, BITTE
   ..."
12 FOR I=828 TO 864:READ A:POKE I,A:PS=PS+
   A+1:NEXT I
13 IF PS<>5802 THEN PRINT"PRUEFSUMMENFEHLE
   R IN ZEILEN 20-22":END
14 SYS 828:PS=0:FOR I=58464 TO 58583:READ
   A:POKE I,A:PS=PS+A+1:NEXT I
15 IF PS<>16267 THEN PRINT"PRUEFSUMMENFEHL
   ER IN ZEILEN 22-30":END
16 POKE 1,53:POKE 42289,96:POKE 42290,228
17 PRINT"{4DOWN,9SPACE}CHECKSUMMER AKTIVIE
   RT."
18 PRINT"{2DOWN}AUSSCHALTEN : POKE1,55"
19 PRINT"{DOWN}ANSCHALTEN{2SPACE}: POKE1,5
   3":NEW
20 DATA 169,0,133,254,162,1,189,93,3,133,2
   55,160,0,177,254
21 DATA 145,254,136,208,249,230,255,165,25
   5,221,95,3,208,238,202
22 DATA 16,230,96,160,224,192,0,160,2,169,
   0,170,133,254,177
23 DATA 95,240,40,201,32,208,3,200,208,245
   ,133,255,138,41,7
24 DATA 170,240,14,72,165,255,24,42,105,0,
   202,208,249,133,255
25 DATA 104,170,232,165,255,24,101,254,133
   ,254,76,111,228,192,4
26 DATA 48,219,198,214,165,214,72,162,3,16
   9,32,157,1,4,189
27 DATA 212,228,32,210,255,208,12,0,92,72,
   32,201,255,170,104
28 DATA 144,1,138,96,202,16,228,166,254,16
   9,0,32,205,189,169
29 DATA 62,32,210,255,104,133,214,32,108,2
   29,169,141,32,210,255
30 DATA 76,128,164,9,60,18,19
```


Der Checksummer 64 V3 erkennt auch Vertauschungen von Zahlen

CTRL steht für Control-Taste, so bedeutet {CTRL-A}, daß Sie die Control-Taste und die Taste »A« drücken müssen. Im folgenden steht:

Steuerbefehl	Taste
{DOWN}	Taste neben rechtem Shift, Cursor unten
{UP}	Shift-Taste & Taste neben rechtem Shift; Cursor hoch
{CLR}	Shift-Taste & 2. Taste ganz rechts oben
{INST}	Shift-Taste & Taste ganz rechts oben
{HOME}	2. Taste von ganz rechts oben
{DEL}	Taste ganz rechts oben
{RIGHT}	Taste ganz rechts unten
{LEFT}	Shift-Taste & Taste unten rechts
{SPACE}	Leertaste
{SHIFT-Space}	Shift-Taste & Leertaste
{F1} bis {F8}	Funktionstasten
{RETURN}	Shift-Taste & Return
{BLACK}	Control-Taste & 1
{WHITE}	Control-Taste & 2
{RED}	Control-Taste & 3
{CYAN}	Control-Taste & 4
{PURPLE}	Control-Taste & 5
{GREEN}	Control-Taste & 6
{BLUE}	Control-Taste & 7
{YELLOW}	Control-Taste & 8
{RVSON}	Control-Taste & 9
{RVOFF}	Control-Taste & 0
{ORANGE}	Commodore-Taste & 1
{BROWN}	Commodore-Taste & 2
{LIG.RED}	Commodore-Taste & 3
{GREY 1}	Commodore-Taste & 4
{GREY 2}	Commodore-Taste & 5
{LIG.GREEN}	Commodore-Taste & 6
{LIG.BLUE}	Commodore-Taste & 7
{GREY 3}	Commodore-Taste & 8

Tabelle 1. Die Steuerbefehle in den Listings

MSE – Abtippen sicher und leicht gemacht

Ähnlich wie der »Checksummer« ist auch der MSE ein Hilfsmittel bei der Eingabe von Listings, diesmal jedoch bei reinen Maschinensprache-programmen.

Im Gegensatz zum »Checksummer« ist die Eingabe von Listings ohne den MSE nicht möglich. Der MSE verringert die Tipparbeit um ein Drittel und schließt Fehleingaben aus. Außerdem können Sie die Werte blind eingeben, ohne andauernd auf den Bildschirm schauen zu müssen. Dies wird durch akustische Meldungen realisiert.

MSE ist ein Maschinenspracheeditor, mit dem ein Vertippen ausgeschlossen ist. Eine abgetippte Zeile wird nur angenommen, wenn sie richtig ist. Eine Checksumme am Ende jeder Zeile prüft, ob die richtigen Werte in der richtigen Zeile an der richtigen Stelle stehen. Wenn nicht, ertönt ein Warnsignal, und man beseitigt den Fehler.

War die Zeile korrekt, erklingt ein Gong, und die nächste Zeilennummer wird ausgegeben. Damit ist also auch »blindes« Eintippen möglich; Sie können sich voll auf den Text konzentrieren.

So arbeitet man mit MSE

Laden und starten Sie MSE. Zuerst wird der Programmname und die Start- und Endadresse erfragt. **Diese Angaben entnehmen Sie dem Kopf des jeweiligen abgedruckten Listings.** MSE meldet sich dann mit der Zeilennummer der ersten Zeile. Wenn Sie die Zeile richtig eingegeben haben, erscheint die nächste Zeilennummer und so weiter bis zum Ende. Zum Schluß wird das fertige Programm mit »CTRL-S« auf Diskette oder Kassette gespeichert. Dazu sind keine weiteren Angaben mehr erforderlich. Das Programm kann dann ganz normal wieder geladen und gestartet werden. Wenn Sie nicht alles auf einmal eingeben wollen, können Sie jederzeit unterbrechen und den eingetippten Teil mit »CTRL-S« abspeichern. Wollen Sie weiterarbeiten, laden und starten Sie MSE wieder.

Geben Sie auf die Frage nach der Startadresse aber jetzt »L« ein, um Ihr Teilprogramm zu laden. Jetzt können Sie mit »CTRL-N« die Adresse eingeben, an der Sie weitertippen müssen. Wenn Sie sich nicht gemerkt haben, wie weit Sie gekommen sind, geben Sie nach dem Laden »CTRL-M« ein.

Auf die Frage nach der Startadresse antworten Sie mit der Anfangsadresse, die links in der Kopfzeile auf dem Bildschirm steht. Nun wird Ihr Programm gelistet. Mit »SPACE« wird das Listen fortgesetzt, mit »STOP« abgebrochen. Das Ende Ihres Programmteils erkennen Sie sehr einfach daran, daß nur noch der Wert »AA« in der Zeile steht. Die Adresse dieser Zeile müssen Sie anschließend mit »CTRL-N« eingeben. Das Programm ist nur mit »STOP/RESTORE« zu verlassen. Speichern Sie aber vorher unbedingt immer Ihren Text ab.

Hinweise zum Abtippen

Vor dem Abtippen oder späteren Wiederladen des MSE-**Laders** müssen Sie unbedingt folgende Zeile eingeben:

POKE 43,1: POKE 44,32: POKE 8192,0: NEW

Den MSE-**Lader** brauchen Sie nur einmal. Nach erfolgreichem Abtippen und Starten mit RUN geht der Lader verloren und es wird das endgültige Programm MSE V1.0 erzeugt. So gehen Sie vor:

Starten Sie das Programm mit RUN. Fehlerhafte Zeilen werden angezeigt und müssen korrigiert werden, bis der Lader zum »READY« durchläuft. Jetzt müssen Sie das fertige MSE-Programm speichern. Dazu brauchen Sie nur »RETURN« zu drücken, weil die erforderlichen Angaben schon auf dem Bildschirm stehen. Datasetten-Besitzer müssen in Zeile 343 die letzte Zahl in »1« abändern. Ab jetzt können Sie »MSE V1.0« direkt, also ohne den DATA-Lader, benutzen. MSE V1.0 wird ganz normal mit »,8« geladen (keine POKEs notwendig).

(N. Mann/D. Weineck/gk)

MSE-Befehle:

DEL	löscht die letzte Eingabe.
CTRL-S	speichert das eingetippte Programm ab.
L oder CTRL-L	lädt ein Programm. Start- und Endadresse werden automatisch ermittelt.
CTRL-M	listet den Speicherinhalt. Abbruch mit STOP-Taste, weiter mit Leertaste.
CTRL-N	erlaubt die Eingabe einer neuen Adresse zum Weitertippen.
CTRL-P	gibt ein MSE-Listing auf dem Drucker aus.

```
100 REM ****************************          <091>
110 REM *                          *          <159>
120 REM *        M S E   LADER     *          <206>
130 REM *                          *          <179>
220 REM ****************************          <211>
230 REM                                       <036>
240 DIM H(75): FOR I=0 TO 9                   <113>
250 H(48+I)=I: H(65+I)=I+10:NEXT              <041>
260 FOR I=2048 TO 3755 : READ A$              <198>
270 H=ASC(LEFT$(A$,1)):L=ASC(RIGHT$(A$,1))    <199>
280 D=H(H)*16+H(L):S=S+D:POKE I,D             <219>
290 A=A+1:IF A<20 THEN NEXT:A=-1              <141>
300 PRINT " ZEILE:";1000+Z;                   <011>
310 READ V :Z=Z+1:IF V=S THEN 330             <218>
320 PRINT"PRUEFSUMMENFEHLER !":STOP           <138>
330 IF A<0 THEN 341                           <221>
340 S=0:A=0:PRINT:NEXT                        <046>
341 PRINT"{CLR}PO43,1:PO44,8:PO45,172:PO46
    ,14                                       <010>
342 POKE 631,19:POKE 632,13:POKE 633,13:PO
    KE 198,3                                  <249>
343 PRINT"{3DOWN}SAVE"CHR$(34)"MSE V1.0"CH
    R$(34)",8                                 <171>
344 END                                       <092>
1000 DATA 00,0B,08,0A,00,9E,32,30,36,31,00
     ,00,00,A2,08,A9,36,85,A4,A9, 1247        <119>
1001 DATA 08,85,A5,A9,00,85,A6,A9,B0,85,A7
     ,A0,00,B1,A4,91,A6,C8,D0,F9, 2888        <054>
1002 DATA E6,A5,E6,A7,CA,D0,F2,A9,36,85,01
     ,4C,00,B0,20,D1,B1,A9,06,8D, 2787        <144>
1003 DATA 21,D0,A9,03,8D,20,D0,8D,86,02,A0
     ,B3,A9,74,20,FF,B1,A0,B3,A9, 2667        <237>
1004 DATA B9,20,FF,B1,A0,00,20,CF,FF,99,01
     ,02,C8,C9,0D,D0,F5,88,F0,D2, 2912        <217>
1005 DATA C0,0F,90,02,A0,0E,8C,00,02,20,EA
     ,B1,A0,B3,A9,CF,20,FF,B1,20, 2323        <013>
1006 DATA 8E,B4,85,FC,85,62,20,8E,B4,85,FB
     ,85,61,20,A7,B4,D0,20,A0,B3, 2864        <199>
1007 DATA A9,E5,20,FF,B1,20,8E,B4,85,60,20
     ,8E,B4,85,5F,20,A7,B4,D0,0A, 2624        <091>
```

Der MSE zum bequemen Abtippen von Maschinenprogrammen

```
1008 DATA A5,61,C5,5F,A5,62,E5,60,90,06,20
     ,43,B3,4C,3A,B0,A9,AA,A0,00, 2379     <167>
1009 DATA 91,FB,E6,FB,D0,02,E6,FC,20,3F,B2
     ,90,EF,4C,FB,B4,A2,02,86,58, 3118     <152>
1010 DATA A9,A6,A0,9D,20,F2,B1,20,E4,FF,F0
     ,FB,C9,30,90,0C,C9,47,B0,08, 2970     <231>
1011 DATA C9,3A,90,0B,C9,41,B0,07,C9,14,D0
     ,0F,4C,0B,B1,20,D2,FF,A6,58, 2322     <121>
1012 DATA 95,F7,C6,58,D0,D2,60,AE,8D,02,F0
     ,26,C9,0C,D0,03,4C,0B,B6,C9, 2685     <057>
1013 DATA 13,D0,03,4C,8B,B5,C9,0D,D0,03,4C
     ,BA,B4,C9,10,D0,03,4C,68,B5, 2282     <225>
1014 DATA C9,0E,D0,06,20,5F,B4,4C,64,B1,4C
     ,92,B0,A5,F9,20,02,B1,0A,0A, 2132     <208>
1015 DATA 0A,0A,85,F9,A5,F8,20,02,B1,05,F9
     ,60,C9,3A,90,02,69,08,29,0F, 1950     <092>
1016 DATA 60,A6,59,E0,08,90,1F,A6,58,E0,02
     ,B0,06,20,D2,FF,4C,8E,B0,C6, 2509     <188>
1017 DATA 59,A0,14,A9,92,20,F2,B1,CA,D0,FA
     ,84,57,68,68,4C,8B,B1,A6,D3, 2891     <197>
1018 DATA E0,08,B0,03,4C,92,B0,20,D2,FF,A6
     ,58,E0,02,90,09,C6,59,20,D2, 2468     <049>
1019 DATA FF,C6,58,D0,F9,4C,8E,B0,48,4A,4A
     ,4A,4A,20,59,B1,68,29,0F,C9, 2419     <035>
1020 DATA 0A,90,02,69,06,69,30,4C,D2,FF,A2
     ,FC,9A,20,D1,B1,20,48,B2,20, 2261     <073>
1021 DATA EA,B1,20,9F,B2,A5,FC,20,4E,B1,A5
     ,FB,20,4E,B1,20,ED,B1,A9,3A, 2860     <148>
1022 DATA A0,20,20,F2,B1,A9,00,85,59,20,8E
     ,B0,20,ED,B1,A4,59,20,EF,B0, 2530     <233>
1023 DATA 91,FB,C8,84,59,C0,08,90,EC,20,10
     ,B2,A9,12,20,D2,FF,20,8E,B0, 2657     <105>
1024 DATA 20,EF,B0,C5,FF,F0,0D,20,43,B3,A9
     ,14,A0,14,20,F2,B1,4C,A2,B1, 2665     <034>
1025 DATA A9,92,20,D2,FF,20,33,B2,20,E0,B2
     ,20,3F,B2,90,9F,4C,8B,B5,A9, 2648     <123>
1026 DATA 93,20,D2,FF,A2,00,A9,03,9D,00,D8
     ,9D,00,D9,9D,00,DA,9D,00,DB, 2476     <237>
1027 DATA E8,D0,EF,60,A9,0D,2C,A9,20,4C,D2
     ,FF,20,D2,FF,98,4C,D2,FF,20, 2965     <160>
1028 DATA E4,FF,F0,FB,60,84,5D,85,5C,A0,00
     ,B1,5C,F0,06,20,D2,FF,C8,D0, 3100     <077>
1029 DATA F6,60,A5,FB,85,5A,A0,00,84,5B,B1
     ,FB,18,65,5A,85,5A,90,02,E6, 2606     <156>
1030 DATA 5B,06,5A,26,5B,C8,C0,08,90,EC,A5
     ,5A,65,5B,85,FF,60,18,A5,FB, 2467     <219>
1031 DATA 69,08,85,FB,90,02,E6,FC,60,A5,FB
     ,C5,5F,A5,FC,E5,60,60,A0,B3, 3106     <183>
1032 DATA A9,FB,20,FF,B1,A0,01,B9,00,02,20
     ,D2,FF,CC,00,02,C8,90,F4,A9, 2692     <098>
1033 DATA 10,ED,00,02,AA,20,ED,B1,CA,D0,FA
     ,A5,62,20,4E,B1,A5,61,20,4E, 2453     <236>
1034 DATA B1,20,ED,B1,A5,60,20,4E,B1,A5,5F
     ,20,4E,B1,A9,9F,20,D2,FF,20, 2575     <038>
1035 DATA EA,B1,24,5E,10,01,60,A9,12,20,D2
     ,FF,A2,28,20,ED,B1,CA,D0,FA, 2646     <161>
1036 DATA A9,92,4C,D2,FF,A5,D6,C9,16,B0,01
     ,60,A9,A0,85,A4,A9,78,85,A6, 2945     <204>
1037 DATA A9,04,85,A5,85,A7,A2,13,A0,27,B1
     ,A4,91,A6,88,10,F9,CA,F0,19, 2671     <208>
1038 DATA 18,A5,A4,69,28,85,A4,90,02,E6,A5
     ,18,A5,A6,69,28,85,A6,90,E0, 2503     <251>
1039 DATA E6,A7,4C,B6,B2,A9,91,4C,D2,FF,A9
     ,0F,8D,18,D4,A9,00,8D,05,D4, 2776     <000>
1040 DATA A9,F7,8D,06,D4,A9,11,8D,04,D4,A9
     ,32,8D,01,D4,A9,00,8D,00,D4, 2413     <126>
1041 DATA A0,80,20,09,B3,A9,10,8D,04,D4,60
     ,A2,FF,CA,D0,FD,88,D0,F8,60, 2914     <240>
1042 DATA A9,0F,8D,18,D4,A9,2D,8D,05,D4,A9
     ,A5,8D,06,D4,A9,21,8D,04,D4, 2385     <119>
1043 DATA A9,07,8D,01,D4,A9,05,8D,00,D4,A0
     ,FF,20,09,B3,A9,20,8D,04,D4, 2250     <078>
1044 DATA A9,00,8D,01,D4,8D,00,D4,60,38,20
     ,F0,FF,8A,48,98,48,18,A0,06, 2179     <175>
1045 DATA A2,18,20,F0,FF,A0,B4,A9,0A,20,FF
     ,B1,20,12,B3,20,E4,FF,F0,FB, 2931     <093>
1046 DATA A2,1D,A9,14,20,D2,FF,CA,D0,FA,68
     ,A8,68,AA,18,4C,F0,FF,0D,0D, 2704     <088>
1047 DATA 0D,20,20,20,20,20,20,20,4D,41,53
     ,43,48,49,4E,45,4E,53,50,52, 1144     <216>
1048 DATA 41,43,48,45,20,2D,20,45,44,49,54
     ,4F,52,20,0D,0D,20,20,20,20, 1023     <038>
1049 DATA 20,20,20,20,56,4F,4E,20,4E,2E,4D
     ,41,4E,4E,20,26,20,44,2E,57, 1128     <206>
1050 DATA 45,49,4E,45,43,4B,00,0D,0D,0D,20
     ,20,20,50,52,4F,47,52,41,4D, 1102     <117>
1051 DATA 4D,4E,41,4D,45,20,3A,20,00,0D,0D
     ,20,20,20,53,54,41,52,54,41, 1073     <095>
1052 DATA 44,52,45,53,53,45,20,3A,20,24,00
     ,0D,0D,20,20,20,45,4E,44,41, 1014     <129>
1053 DATA 44,52,45,53,53,45,20,20,20,3A,20
     ,24,00,92,05,20,50,52,4F,47, 1171     <217>
1054 DATA 52,41,4D,4D,20,3A,20,00,12,20,20
     ,2A,2A,2A,20,46,41,4C,53,43, 1024     <027>
1055 DATA 48,45,20,45,49,4E,47,41,42,45,20
     ,2A,2A,2A,20,20,92,00,0D,0D, 1058     <098>
1056 DATA 2A,2A,2A,20,45,4E,44,45,20,2A,2A
     ,2A,00,13,05,20,20,12,44,92, 920      <148>
1057 DATA 49,53,4B,20,4F,44,45,52,20,12,54
     ,92,41,50,45,0D,00,13,20,20, 1151     <035>
1058 DATA 49,2F,4F,20,2D,20,46,45,48,4C,45
     ,52,00,20,D1,B1,20,48,B2,A0, 1606     <012>
1059 DATA B3,A9,CF,20,FF,B1,20,8E,B4,85,FC
     ,20,8E,B4,85,FB,C5,61,A5,FC, 3207     <251>
1060 DATA E5,62,90,23,A5,FB,C5,5F,A5,FC,E5
     ,60,B0,19,20,A7,B4,D0,14,60, 2860     <112>
1061 DATA 20,A7,B4,F0,0C,85,F9,20,A7,B4,F0
     ,05,85,F8,4C,EF,B0,68,68,20, 2749     <088>
1062 DATA 43,B3,4C,5F,B4,20,CF,FF,C9,4C,D0
     ,09,20,D1,B1,20,48,B2,4C,0B, 2372     <046>
1063 DATA B6,C9,0D,60,A9,00,85,5E,20,5F,B4
     ,20,EA,B1,20,0D,B5,24,5E,30, 2042     <120>
1064 DATA 05,20,E4,FF,F0,FB,20,E1,FF,F0,26
     ,20,9F,B2,24,5E,10,09,20,4E, 2435     <198>
1065 DATA B5,20,0D,B5,20,60,B5,20,33,B2,20
     ,3F,B2,90,D7,A0,B4,A9,28,20, 2190     <207>
1066 DATA FF,B1,20,E4,FF,C9,0D,D0,F9,A9,00
     ,85,5E,A5,61,85,FB,A5,62,85, 3056     <240>
1067 DATA FC,20,E0,B2,4C,64,B1,A5,FC,20,4E
     ,B1,A5,FB,85,FF,20,4E,B1,A9, 3003     <221>
1068 DATA 20,A0,3A,20,F2,B1,A0,00,20,ED,B1
     ,B1,FB,20,4E,B1,C8,C0,08,90, 2566     <070>
1069 DATA F3,20,ED,B1,24,5E,30,03,A9,12,2C
     ,A9,20,20,D2,FF,20,10,B2,A5, 2190     <059>
1070 DATA FF,20,4E,B1,A9,92,20,D2,FF,4C,EA
     ,B1,A9,FF,85,B8,85,B9,A9,04, 3073     <029>
1071 DATA 85,BA,20,C0,FF,A2,FF,4C,C9,FF,20
     ,CC,FF,A9,FF,4C,C3,FF,20,5F, 3315     <189>
1072 DATA B4,A9,80,85,5E,20,4E,B5,20,48,B2
     ,A2,24,A9,2D,20,D2,FF,CA,D0, 2596     <111>
1073 DATA FA,20,EA,B1,20,EA,B1,20,60,B5,4C
     ,C1,B4,20,B8,B5,A6,5F,A4,60, 2812     <015>
1074 DATA A9,61,20,D8,FF,B0,0A,20,B7,FF,29
     ,BF,D0,03,4C,FB,B4,A9,01,20, 2577     <201>
1075 DATA C3,FF,20,68,B6,A0,B4,A9,4F,20,FF
     ,B1,20,F9,B1,4C,FB,B4,20,68, 2921     <237>
1076 DATA B6,A9,37,A0,B4,20,FF,B1,20,F9,B1
     ,A2,08,C9,44,F0,06,A2,01,C9, 2717     <213>
1077 DATA 54,D0,F1,A9,01,A8,20,BA,FF,A0,00
     ,E0,01,F0,1A,A9,40,8D,20,02, 2403     <101>
1078 DATA A9,3A,8D,21,02,B9,01,02,99,22,02
     ,C8,CC,00,02,90,F4,C8,C8,D0, 2182     <127>
1079 DATA 0C,B9,01,02,99,20,02,C8,CC,00,02
     ,D0,F4,98,A2,20,A0,02,4C,BD, 2018     <025>
1080 DATA FF,20,B8,B5,A5,BA,C9,08,90,33,A6
     ,B9,86,57,A9,01,20,C3,FF,A9, 2800     <022>
1081 DATA 60,85,B9,20,C0,FF,B0,28,A5,BA,20
     ,B4,FF,A5,B9,20,96,FF,20,A5, 2911     <053>
1082 DATA FF,85,61,A5,90,4A,4A,B0,13,20,A5
     ,FF,85,62,20,AB,FF,A5,57,85, 2663     <214>
1083 DATA B9,A9,00,20,D5,FF,90,03,4C,A3,B5
     ,86,5F,84,60,A5,BA,C9,01,D0, 2639     <131>
1084 DATA 0A,AD,3D,03,85,61,AD,3E,03,85,62
     ,4C,FB,B4,A9,13,20,D2,FF,A2, 2300     <120>
1085 DATA 1C,20,ED,B1,CA,D0,FA,60, 1230   <214>
```


MSE (Schluß). Dieses Listing können Sie (müssen aber nicht) mit dem Checksummer 64 V3 in diesem Heft eingeben.

Schnell kopiert mit Hypra-Copy

Hypra-Copy ist ein schnelles und komfortables Filecopy-Programm für den C64. Das Kopieren wird um das Vier- bis Fünffache beschleunigt.

Trotz der aufwendigen Lade- und Speicherroutinen ist es gelungen, das Programm insgesamt recht kurz zu halten; Hypra-Copy belegt, gespeichert auf der Diskette, nur ganze 15 Blöcke. Somit ist der verfügbare Arbeitsspeicher sehr groß: Hypra-Copy kann sich bis zu 30 Filenamen »merken«, und in einem Ladegang können maximal 232 Blöcke eingelesen werden. Hat man das Programm (siehe Listing) mit dem MSE abgetippt und gespeichert, kann es später ganz normal mit »LOAD "HYPRA-COPY",8« geladen und mit »RUN« gestartet werden. Es erscheint dann das Hauptmenü:

HYPRA-COPY

:C: Copy Files (kopiere Programme)
:S: Scratch Files (lösche Programme)
:D: Directory (Inhaltsverzeichnis der Diskette)
:O: Order Disk (Befehl an Floppy)

Die Bedienung ergibt sich damit eigentlich schon von selbst. Drückt man die Taste »O«, erscheint auf dem Bildschirm eine eckige Klammer mit einem blinkenden Cursor dahinter.

Nun kann man den Befehl eingeben, der (normalerweise mit »OPEN 1,8,15,"Befehl":CLOSE 1«) zur Floppy geschickt werden soll. Reagiert die Floppy mit einer Fehlermeldung, so wird diese auf dem Bildschirm ausgegeben.

Bei Betätigung der Taste »D« erscheint das Directory. Die Anzeige auf dem Bildschirm kann durch die CTRL-Taste angehalten werden.

Durch Drücken der »S«-Taste gelangt man in den Scratch-Modus (löschen von Files). Um Verwechslungen mit dem Copy-Modus auszuschließen, wird erst einmal mit einer dicken, reversen Balkenüberschrift darauf aufmerksam gemacht, daß man sich tatsächlich im Scratch-Modus befindet. Danach erscheint das Directory, jedoch erscheint hinter jedem Filenamen ein »(Y/N)«; relative Files sowie Files, deren Länge gleich 0 oder größer als 232 Blöcke ist, werden bei dieser Befragung übergangen, denn diese Routine wird auch beim Kopieren gebraucht, und Files mit diesen Eigenschaften kann Hypra-Copy nicht kopieren.

Programme löschen

Files, die sich Hypra-Copy merken soll, also die, die gelöscht werden sollen, müssen mit »Y« markiert werden. Ist man nur versehentlich im Scratch-Modus gelandet, so kann man diesen Modus mit der STOP-Taste verlassen (ansonsten dient die STOP-Taste dazu, Hypra-Copy zu beenden; man kann es dann aber wieder mit »RUN« starten).

So werden nach und nach alle Files des Directories durchgegangen. Will man nur einige wenige Files am Anfang einer ellenlangen Directory löschen (oder kopieren), so braucht man die restlichen Files nicht mehr mit »N« zu bearbeiten, es genügt ein Druck auf die »↑«-Taste, und man gelangt zum Ende des Directory. Hier wird dann gefragt, ob man nun auch sicher ist, daß diese Files gelöscht werden sollen. Beantwortet man diese Frage mit »N«, gelangt man zurück in das Hauptmenü, andernfalls werden die markierten Files gelöscht.

Damit man weiß, wie weit das Programm mit dem Löschen ist, wird immer der Name des Files ausgegeben, das gerade gelöscht wird. Am Ende des Löschvorgangs gelangt man automatisch wieder in das Hauptmenü.

Der Copy-Modus

So, nun zum Copy-Modus, in den man mit der »C«-Taste kommt. Zuerst werden wieder, genau wie beim Löschen, die Namen der zu kopierenden Files eingelesen.

Danach wird gefragt, ob man die Files einzeln oder gesammelt kopieren will und ob beim Speichern ein »VERIFY« durchgeführt werden soll. Verzichtet man auf dieses Verify, so wird etwas schneller kopiert. Um den Bedienungskomfort zu erhöhen, werden die Antworten auf diese Fragen, wie auch auf die »SURE?«-Frage (sure = Sind Sie sicher?) im Scratch-Modus und die »SAVE BUFFER AGAIN?«-Frage (Soll das gleiche noch einmal gespeichert werden?), nicht per GET ($ffe4), sondern per BASIN ($ffcf) eingelesen, wobei die wohl gebräuchlichere Antwort schon vorgegeben ist; man braucht also nur noch RETURN drücken; natürlich kann diese vorgegebene Antwort aber auch überschrieben werden.

Nun werden die Programme geladen. Ist die Summe der Blöcke der einzelnen Files kleiner als 233, können sogar alle auf einmal eingelesen werden; andernfalls wird ein erneuter Ladeanlauf nötig.

Sind die Files geladen, wird man aufgefordert, die Ziel-Diskette, auf die kopiert werden soll, einzulegen; außerdem wird noch die Anzahl der zu speichernden Blocks angegeben, und ein Untermenü erscheint.

Es besteht jetzt noch die Möglichkeit, diverse Directories anzusehen oder Befehle zur Floppy zu schicken.

Man kann also in aller Ruhe eine passende Diskette aussuchen. Ist man gewillt fortzufahren, so drückt man einfach die Funktionstaste F7, dann beginnt nämlich das Speichern. Anschließend wird man gefragt, ob man eben diese Files noch einmal auf eine andere Diskette schreiben will. Beantwortet man diese Frage mit »Y«, beginnt der komplette SAVE-Vorgang von neuem. Andernfalls wird normal fortgefahren und, wie schon erwähnt, können erneute Ladeanläufe erforderlich werden.

Wurden alle Files kopiert, so wird in das Hauptmenü zurückgekehrt.

Erste Hilfe bei Fehlern

Nun zur Fehlerbehandlung: Mit »Fehlern« sind nicht Fehler gemeint, die in der Programmstruktur von Hypra-Copy liegen, sondern Fehler, die von der Floppy signalisiert werden; sei es, daß man einen unkorrekten Befehl an die Floppy schickt, beim Kopieren die falsche Diskette einlegt oder Write- oder Read-Errors auftreten.

In solchen oder ähnlichen Fällen besteht immer die Mög-

lichkeit, den Vorgang, bei dem der Fehler aufgetreten ist, zu wiederholen beziehungsweise zu übergehen. Treten Fehler beim Öffnen eines Files auf, so kann man sogar nochmals Directories ansehen und Befehle an die Floppy schicken. War es nicht möglich, ein File korrekt zu laden und wurde es übergangen, so wird dies bei der Angabe der zu speichernden Blöcke, wie auch beim Speichern selbst, berücksichtigt. Können Files aus irgendeinem Grund nicht korrekt gespeichert werden, so kann man diese auch überspringen.

(Burkhard Graves/og)

Hinweis zum Abtippen
Dieses Listing muß mit dem MSE eingegeben werden.
Den MSE finden Sie in dieser Ausgabe auf Seite 136.

```
programm : hypra-copy          0801 16b3
-----------------------------------------

0801 : 0b 08 c1 07 9e 32 30 36  0a
0809 : 31 00 00 00 a9 0b 8d 20  a4
0811 : d0 8d 21 d0 78 a9 c1 8d  01
0819 : 18 03 58 a2 00 86 05 8e  83
0821 : 3b 09 a9 40 85 9d a9 d5  eb
0829 : a0 08 20 1e ab 20 e1 09  ef
0831 : 4c 1c 08 a5 f7 18 69 13  4e
0839 : 85 f7 90 02 e6 f8 c9 ec  55
0841 : 60 a9 b2 85 f7 85 f9 a9  ba
0849 : 16 85 f8 85 fa 60 0d 12  1c
0851 : 20 21 20 53 43 52 41 54  e9
0859 : 43 48 20 46 49 4c 45 53  44
0861 : 20 21 20 11 0d 00 0d 11  63
0869 : 43 4f 50 59 20 53 45 50  e5
0871 : 41 52 41 54 45 4c 59 20  13
0879 : 3f 20 4e 9d 00 0d 11 56  69
0881 : 45 52 49 46 59 20 3f 20  de
0889 : 4e 9d 00 0d 11 53 55 52  ed
0891 : 45 20 3f 20 59 9d 00 0d  57
0899 : 11 3a 54 3a 20 54 52 59  c4
08a1 : 20 41 47 41 49 4e 0d 11  b9
08a9 : 3a 46 37 3a 20 54 4f 20  3e
08b1 : 43 4f 4e 54 49 4e 55 45  a1
08b9 : 11 0d 00 0d 11 53 41 56  50
08c1 : 45 20 42 55 46 46 45 52  a2
08c9 : 20 41 47 41 49 4e 20 3f  8a
08d1 : 20 4e 9d 00 93 05 11 1d  60
08d9 : 3e 20 48 59 50 52 41 2d  5c
08e1 : 43 4f 50 59 20 3c 0d 1d  5d
08e9 : 1d c4 c4 c4 c4 c4 c4 c4  41
08f1 : c4 c4 c4 c4 c4 11 11 0d  14
08f9 : 3a 43 3a 20 43 4f 50 59  0a
0901 : 20 46 49 4c 45 53 11 0d  6e
0909 : 3a 53 3a 20 53 43 52 41  9b
0911 : 54 43 48 20 46 49 4c 45  87
0919 : 53 0d 11 3a 44 3a 20 44  9e
0921 : 49 52 45 43 54 4f 52 59  09
0929 : 11 0d 3a 4f 3a 20 4f 52  c0
0931 : 44 45 52 20 44 49 53 4b  23
0939 : 11 0d 00 91 3a 46 37 3a  2a
0941 : 20 54 4f 20 43 4f 4e 54  f4
0949 : 49 4e 55 45 11 0d 00 12  55
0951 : 28 59 2f 4e 29 92 00 14  0b
0959 : 14 14 14 14 9d 3c 2d 00  6f
0961 : 0d 11 4c 49 53 54 20 46  18
0969 : 55 4c 4c 20 21 11 0d 00  ca
0971 : 0d 49 4e 53 45 52 54 20  99
0979 : 53 4f 55 52 43 45 20 44  7b
0981 : 49 53 4b 0d 00 0d 49 4e  13
0989 : 53 45 52 54 20 54 41 52  ec
0991 : 47 45 54 20 44 49 53 4b  06
0999 : 20 c3 20 00 20 42 4c 4f  87
09a1 : 43 4b 53 20 54 4f 20 53  4a
09a9 : 41 56 45 0d 00 0d 11 11  d7
09b1 : 44 49 53 4b 20 45 52 52  f2
09b9 : 4f 52 20 00 0d 11 11 4c  70
09c1 : 4f 41 44 20 45 52 52 4f  95
09c9 : 52 20 21 00 20 42 08 a0  e9
09d1 : 00 98 91 f7 c8 d0 fb e6  51
09d9 : f8 a6 f8 e0 20 d0 f3 60  98
09e1 : a9 00 85 c6 20 e4 ff f0  cf
09e9 : fb a6 05 f0 03 4c c9 0a  65
09f1 : c9 43 d0 4a 20 cd 09 a2  b3
09f9 : 00 86 03 20 42 08 20 34  4f
0a01 : 0b a2 00 bd b2 16 d0 01  36
0a09 : 60 a9 67 a0 08 20 1e ab  7d
0a11 : a9 00 85 c6 85 41 20 cf  77
0a19 : ff c9 59 d0 02 85 41 a9  12
0a21 : 4c 8d ab 15 a9 7e a0 08  e2
0a29 : 20 1e ab a9 00 85 c6 20  00
0a31 : cf ff c9 59 d0 05 a9 2c  d2
0a39 : 8d ab 15 4c 02 0d c9 53  c1
0a41 : f0 03 4c c9 0a a9 a4 a0  c1
0a49 : 13 20 2d 0b a9 4f a0 08  c1
0a51 : 20 1e ab a9 00 85 03 20  19
0a59 : cd 09 20 42 08 20 34 0b  64
0a61 : 20 42 08 a9 00 85 c6 a9  74
0a69 : 8c a0 08 20 1e ab 20 cf  ab
0a71 : ff c9 59 f0 01 60 a9 0d  9d
0a79 : 20 d2 ff 20 d2 ff a9 08  ea
0a81 : 20 b1 ff a9 6f 20 93 ff  f5
0a89 : a0 00 b1 f7 f0 35 a9 53  9b
0a91 : 20 a8 ff 20 d2 ff a9 3a  51
0a99 : 20 a8 ff 20 d2 ff c8 b1  c5
0aa1 : f7 c9 2c f0 09 20 a8 ff  da
0aa9 : 20 d2 ff 4c 9f 0a a9 0d  c7
0ab1 : 20 a8 ff 20 d2 ff a9 08  0d
0ab9 : 20 ae ff 20 c9 0c 20 34  1a
0ac1 : 08 d0 bb a9 08 4c ae ff  f3
0ac9 : c9 88 d0 01 60 c9 54 d0  72
0ad1 : 01 60 c9 44 d0 0e a2 19  37
0ad9 : 86 03 20 34 0b a6 cb e0  46
0ae1 : 40 f0 fa 60 c9 4f d0 31  21
0ae9 : a9 5d 20 d2 ff a9 08 20  51
0af1 : b1 ff a9 6f 20 93 ff a9  ec
0af9 : 00 85 c6 20 cf ff 20 a8  40
0b01 : ff c9 0d d0 f6 20 d2 ff  fe
0b09 : a9 08 20 ae ff 20 c9 0c  d4
0b11 : b0 01 60 20 69 0c 90 d0  39
0b19 : 60 c9 03 d0 03 4c bb 0c  d2
0b21 : 4c e5 09 a9 0d 20 d2 ff  f4
0b29 : a0 27 a9 c3 20 d2 ff 88  e9
0b31 : d0 fa 60 20 47 0c a9 60  d7
0b39 : 85 b9 a9 01 a2 d0 a0 ff  58
0b41 : 20 8d 0c 90 03 4c 60 0c  69
0b49 : a5 ba 20 b4 ff a5 b9 20  3e
0b51 : 96 ff a9 00 85 90 a0 03  b7
0b59 : 20 4f 0c 85 02 20 4f 0c  4b
0b61 : 88 d0 f5 84 05 a6 02 20  2d
0b69 : cd bd a9 20 20 d2 ff 20  5c
0b71 : 4f 0c a4 03 d0 42 c9 22  da
0b79 : d0 3e 85 05 20 d2 ff a0  44
0b81 : 13 a9 00 85 06 91 f7 88  f7
0b89 : 10 fb c8 20 4f 0c 20 d2  48
0b91 : ff c9 22 f0 05 c8 91 f7  e8
0b99 : d0 f1 a9 2c c8 91 f7 20  8b
0ba1 : 4f 0c c9 50 f0 08 c9 53  90
0ba9 : f0 04 c9 55 d0 05 c8 91  34
0bb1 : f7 d0 05 20 d2 ff d0 e7  96
0bb9 : aa f0 05 20 d2 ff d0 af  f0
0bc1 : a5 02 f0 73 c9 e9 b0 6f  9f
0bc9 : a5 05 f0 6b a9 1d 85 d3  dc
0bd1 : a9 50 a0 09 20 1e ab a9  e1
0bd9 : 00 85 c6 85 fb 20 e4 ff  52
0be1 : f0 fb c9 03 d0 08 20 47  fe
0be9 : 0c 68 68 4c 1c 08 c9 5e  b3
0bf1 : d0 05 85 06 4c 24 0c c9  10
0bf9 : 59 d0 24 a9 9d 8d 5d 09  c6
0c01 : 20 19 0c a0 00 a5 02 91  1d
0c09 : f7 20 34 08 d0 29 a9 61  de
0c11 : a0 09 20 1e ab 4c 47 0c  54
0c19 : a9 58 a0 09 4c 1e ab c9  30
0c21 : 4e d0 ba a9 00 8d 5d 09  af
0c29 : 20 19 0c a0 00 98 91 f7  e8
0c31 : a5 06 f0 03 4c 47 0c a9  f8
0c39 : 0d 20 d2 ff ad 8d 02 d0  fc
0c41 : fb a0 02 4c 59 0b a9 08  3b
0c49 : 20 c3 ff 4c e7 ff 20 a5  1e
0c51 : ff a6 90 f0 12 68 68 20  2c
0c59 : 47 0c 20 c9 0c 90 08 20  8d
0c61 : 69 0c b0 03 4c 34 0b 60  b0
0c69 : a9 98 a0 08 20 1e ab a0  6a
0c71 : 00 84 c6 20 e4 ff aa a9  b5
0c79 : 00 e0 54 d0 02 18 60 e0  3d
0c81 : 88 d0 02 38 60 e0 03 d0  b4
0c89 : ea 4c bb 0c 20 bd ff 20  3a
0c91 : c1 f5 a9 08 85 b8 85 ba  62
0c99 : a9 08 20 b1 ff a9 6f 20  d0
0ca1 : 93 ff a9 55 20 a8 ff a9  e3
0ca9 : 49 20 a8 ff a9 08 20 ae  e5
0cb1 : ff 20 c0 ff b0 03 4c c9  d8
0cb9 : 0c 60 20 47 0c 20 15 fd  f8
0cc1 : a9 0d 20 d2 ff 4c 9d e3  f4
0cc9 : a9 08 20 b4 ff a9 6f 20  60
0cd1 : 96 ff 20 a5 ff aa 38 e9  2e
0cd9 : 30 85 8c f0 0a 8a 48 a9  76
0ce1 : ae a0 09 20 1e ab 68 a6  54
0ce9 : 8c f0 03 20 d2 ff 20 a5  ab
0cf1 : ff c9 0d d0 f2 a9 08 20  0f
0cf9 : ab ff 18 a5 8c f0 01 38  23
0d01 : 60 20 42 08 a9 00 85 8b  cb
0d09 : a9 00 85 ae 85 06 a8 99  48
0d11 : 55 16 c8 c0 5a d0 f8 a9  1f
0d19 : 19 85 af a0 00 b1 f7 d0  04
0d21 : 01 60 20 24 0b a5 8b f0  cd
0d29 : 10 a9 71 a0 09 20 1e ab  e0
0d31 : a9 00 85 c6 20 e4 ff f0  1f
0d39 : fb a9 e8 85 40 85 fd e6  ea
0d41 : 8b a0 00 b1 f7 85 fe f0  dc
0d49 : 72 c5 40 90 06 20 bc 0d  2e
0d51 : 4c 09 0d a2 00 86 b9 a5  20
0d59 : 40 38 e5 fe 85 40 a5 fd  fb
0d61 : 38 e5 fe 85 fd a5 ae 85  cf
0d69 : 69 a5 af 85 6a a5 69 85  c6
0d71 : ae a5 6a 85 af 20 5c 0e  c7
0d79 : b0 11 20 cc 0e 90 17 a9  69
0d81 : bd a0 09 20 1e ab 20 69  67
0d89 : 0c 90 e2 a5 fd 18 65 fe  7f
0d91 : 85 fd a9 00 f0 02 a9 ea  1b
0d99 : 85 ff a4 06 a5 af 99 55  f1
0da1 : 16 c8 a5 ae 99 55 16 c8  89
0da9 : a5 ff 99 55 16 c8 84 06  25
0db1 : 20 34 08 f0 06 a5 41 d0  40
0db9 : 94 f0 86 a9 00 85 c1 85  da
0dc1 : 06 a9 19 85 c2 a5 f9 85  df
0dc9 : f7 a5 fa 85 f8 20 24 0b  39
0dd1 : a9 86 a0 09 20 1e ab a9  fc
0dd9 : e8 38 e5 fd aa a9 00 20  4f
0de1 : cd bd a9 9d a0 09 20 1e  ba
0de9 : ab a9 0d 8d 3b 09 85 05  7a
0df1 : a9 1b a0 09 20 1e ab 20  53
0df9 : e1 09 c9 88 d0 eb a4 06  ed
0e01 : b9 55 16 f0 38 85 af c8  08
0e09 : b9 55 16 85 ae c8 b9 55  66
0e11 : 16 c8 84 06 aa f0 19 a9  57
0e19 : 01 85 b9 20 5c 0e b0 10  68
0e21 : 20 8f 12 20 47 0c 20 c9  7a
0e29 : 0c 90 05 20 69 0c 90 eb  d4
0e31 : a5 ae 85 c1 a5 af 85 c2  3a
0e39 : 20 34 08 d0 c1 a9 bc a0  2d
0e41 : 08 20 1e ab a9 00 85 c6  95
0e49 : 20 cf ff c9 59 d0 03 4c  4b
0e51 : bc 0d a5 f7 85 f9 a5 f8  ad
0e59 : 85 fa 60 a2 05 a5 b9 29  7e
0e61 : 01 d0 07 a2 20 a0 49 4c  a5
0e69 : 6d 0e a0 51 20 2f f1 8a  88
0e71 : 20 d2 ff a6 f7 a4 f8 e8  29
0e79 : d0 01 c8 a9 12 20 8d 0c  a2
0e81 : b0 0c a9 0b a0 4f 99 00  ee
0e89 : d8 88 10 fa 18 60 20 47  9c
0e91 : 0c a5 b9 85 fe a9 00 8d  e7
0e99 : 3b 09 a9 0d 20 d2 ff 85  08
0ea1 : 05 a9 1b a0 09 20 1e ab  b7
0ea9 : a9 91 20 d2 ff 20 d2 ff  c9
0eb1 : a9 98 a0 08 20 1e ab 20  b1
0eb9 : e1 09 c9 54 d0 07 a5 fe  f6
0ec1 : 85 b9 4c 5c 0e c9 88 d0  b4
0ec9 : d1 38 60 a5 ba 20 b4 ff  03
0ed1 : a5 b9 20 96 ff a0 00 20  73
0ed9 : a5 ff a2 34 86 01 91 ae  c1
0ee1 : a2 37 86 01 c8 c0 02 d0  1d
0ee9 : ee 20 47 0c a5 ae 18 69  3e
0ef1 : 02 85 ae 90 0b e6 af d0  bc
0ef9 : 07 a5 a7 c9 08 f0 17 00  5a
0f01 : a9 01 85 a7 a9 00 85 90  53
0f09 : a5 a7 20 0c ed a9 6f 20  36
0f11 : b9 ed a5 90 10 e3 e6 a7  47
0f19 : a5 a7 c9 10 d0 e6 a9 8f  10
0f21 : a2 10 85 a7 86 a8 a9 00  76
0f29 : a2 03 85 a9 86 aa a9 08  58
0f31 : 20 0c ed a9 6f 20 b9 ed  c3
0f39 : a9 4d 20 dd ed a9 2d 20  6e
0f41 : dd ed a9 57 20 dd ed a0  54
0f49 : 00 a5 a9 20 dd ed a5 aa  c3
0f51 : 20 dd ed a9 1e 20 dd ed  47
0f59 : b1 a7 20 dd ed c8 c0 1e  06
0f61 : 90 f6 20 fe ed 18 a5 a7  da
0f69 : 69 1e 85 a7 90 03 e6 a8  46
0f71 : 18 a5 a9 a6 aa 69 1e 85  14
0f79 : a9 90 02 e6 aa e0 05 90  af
0f81 : ad c9 00 90 a9 a9 08 20  6d
0f89 : 0c ed a9 6f 20 b9 ed a9  bf
0f91 : 4d 20 dd ed a9 2d 20 dd  64
0f99 : ed a9 45 20 dd ed a9 8b  bb
0fa1 : 20 dd ed a9 04 20 dd ed  f5
0fa9 : a9 0b 8d 11 d0 20 fb ed  37
0fb1 : ea ea 78 a9 01 85 a7 a0  80
0fb9 : ff 20 39 10 c0 ff f0 44  71
0fc1 : a2 34 86 01 a2 02 a5 a7  5f
0fc9 : f0 02 a2 04 ad 2e 03 d0  dd
0fd1 : 07 ee 2f 03 ad 2f 03 2c  34
0fd9 : a9 00 85 a8 bd 2e 03 91  75
0fe1 : ae e6 ae d0 02 e6 af e8  b0
0fe9 : e4 a8 d0 f0 a2 00 86 a7  07
0ff1 : a2 37 86 01 ad 2e 03 d0  eb
0ff9 : be a9 1b 8d 11 d0 a9 40  c3
1001 : 85 90 18 60 a9 1b 8d 11  ac
1009 : d0 ea a9 1d 38 60 a9 0b  a0
1011 : 8d 00 dd 2c 00 dd 10 fb  c2
1019 : a9 03 8d 00 dd a2 05 ca  44
1021 : ea d0 fc a2 04 ad 00 dd  70
1029 : 2a 2a 66 b0 6a 66 b0 ea  8a
1031 : ca d0 f2 a5 b0 49 ff 60  eb
1039 : 20 0f 10 c9 ff f0 f8 a0  cb
1041 : 00 a9 0b 8d 00 dd 2c 00  2a
1049 : dd 10 fb a9 03 8d 00 dd  bb
1051 : a2 07 ca d0 fd ad 00 dd  4c
1059 : 2a 2a 66 b0 6a 66 b0 ea  ba
1061 : ea ad 00 dd 2a 2a 66 b0  cd
1069 : 6a 66 b0 ea ea ad 00 dd  68
1071 : 2a 2a 66 b0 6a 66 b0 ea  d2
1079 : ea ad 00 dd 2a 2a 66 b0  e5
1081 : 6a 66 b0 a5 b0 49 ff 99  88
1089 : 2e 03 c8 d0 b4 60 a5 00  6a
1091 : 29 06 c9 02 f0 03 4c 9e  06
1099 : fd ea a9 05 85 09 a2 5a  f6
10a1 : 86 4b a2 00 a9 52 85 24  01
10a9 : 20 56 f5 50 fe b8 ad 01  ea
10b1 : 1c c5 24 f0 09 c6 4b d0  6d
10b9 : ef a9 0a 4c 69 f9 50 fe  2f
10c1 : b8 ad 01 1c 95 25 e8 e0  fc
10c9 : 07 d0 f3 20 97 f4 a5 16  1d
10d1 : 45 17 45 18 45 19 45 1a  5d
10d9 : f0 07 c6 09 d0 c0 4c 1e  a0
10e1 : f4 a5 18 c5 06 f0 03 4c  f3
10e9 : 0b f4 85 22 a9 06 85 31  57
10f1 : 4c 3c 04 a5 12 a6 13 85  bf
10f9 : [illegible] 86 17 a5 06 85 18 a5  05
1101 : [illegible] 85 19 a9 00 45 16 45  53
1109 : 17 45 18 45 19 85 1a 20  d8
1111 : 34 f9 a2 5a 20 56 f5 a0  04
1119 : 00 50 fe b8 ad 01 1c d9  1f
1121 : 24 00 f0 06 ca d0 ed 4c  c6
1129 : 51 f5 c8 c0 08 d0 ea 20  b2
1131 : 56 f5 50 fe b8 ad 01 1c  ab
1139 : 91 30 c8 d0 f5 a0 ba 50  1e
1141 : fe b8 ad 01 1c 99 00 01  b7
1149 : c8 d0 f4 20 e0 f8 a5 38  97
1151 : c5 47 f0 03 4c f6 f4 20  e7
1159 : e9 f5 c5 3a f0 03 4c 02  52
1161 : f5 a0 00 a9 55 20 52 04  83
1169 : b9 00 06 85 77 2c 00 18  5e
1171 : 10 fb a9 10 8d 00 18 2c  7d
1179 : 00 18 30 fb a2 00 8a 66  32
1181 : 77 2a 2a 66 77 2a 2a 8d  f1
1189 : 00 18 8a 66 77 2a 2a 66  43
1191 : 77 2a 2a 8d 00 18 8a 66  11
1199 : 77 2a 2a 66 77 2a 2a 8d  09
11a1 : 00 18 8a 66 77 2a 2a 66  5b
11a9 : 77 2a 2a 8d 00 18 a2 02  c1
11b1 : ca d0 fd a9 0f 8d 00 18  25
11b9 : c8 d0 ad ea ea ea ea ea  39
11c1 : ea ea ad 00 1c 09 08 8d  d1
11c9 : 00 1c ad 00 06 d0 03 4c  ce
11d1 : 9e fd c5 18 d0 f9 85 06  e1
11d9 : ad 01 06 85 07 4c 65 03  a8
11e1 : 85 77 2c 00 18 10 fb a9  72
11e9 : 10 8d 00 18 2c 00 18 30  47
11f1 : fb a2 04 a9 00 66 77 2a  d9
11f9 : 2a 66 77 2a 2a 8d 00 18  b9
1201 : ca d0 f0 ea ea ea ea ea  54
1209 : ea a9 0f 8d 00 18 60 60  40
1211 : 85 00 58 a5 00 30 fc 78  c8
1219 : 60 78 ea ea ea ea ea ea  55
1221 : a5 18 8d 00 06 85 06 a5  26
1229 : 19 8d 01 06 85 07 a9 04  49
1231 : 85 78 a9 e2 20 82 04 c9  73
1239 : 02 90 33 a0 00 84 78 a4  b4
1241 : 78 b9 db fe f0 12 58 20  ae
1249 : 76 d6 78 a9 e2 20 82 04  bf
1251 : c9 02 90 1a e6 78 d0 e7  c8
1259 : a9 c0 20 82 04 a9 e2 20  14
1261 : 82 04 c9 02 90 08 a9 ff  88
1269 : 20 52 04 4c 22 eb ad 00  75
1271 : 06 f0 f8 c5 18 f0 c4 ad  5e
1279 : 00 06 85 06 ad 01 06 85  a4
1281 : 07 4c a0 04 ea ea ea ea  de
1289 : a0 00 b9 19 f4 99 ad 00  8e
1291 : dd 8d 1f 01 a5 ba 20 0c  e6
1299 : ed a5 b9 20 b9 ed a9 00  7d
12a1 : 85 90 20 dd ed 20 dd ed  65
12a9 : 20 fe ed a5 90 f0 02 18  41
12b1 : 60 a9 20 a2 14 85 ac 86  6f
12b9 : ad a9 46 a2 01 85 14 86  ba
12c1 : 15 a2 05 a5 ba 20 0c ed  d6
12c9 : a9 6f 20 b9 ed a0 00 b9  c0
12d1 : 18 14 20 dd ed c8 c0 03  e5
12d9 : d0 f5 a5 14 20 dd ed a5  84
12e1 : 15 20 dd ed a9 1e 20 dd  03
12e9 : ed a0 00 b1 ac 20 dd ed  7c
12f1 : c8 c0 1e 90 f6 20 fe ed  fb
12f9 : a5 ac 18 69 1e 85 ac 90  09
1301 : 02 e6 ad a5 14 18 69 1e  7a
1309 : 85 14 90 02 e6 15 ca d0  e1
1311 : b2 a5 ba 20 0c ed a9 6f  fe
1319 : 20 b9 ed a0 00 b9 1b 14  08
1321 : 20 dd ed c8 c0 05 d0 f5  28
1329 : a0 00 20 fe ed 78 a2 00  de
1331 : 8e 00 dd ad 11 d0 29 ef  08
1339 : 8d 11 d0 bd b4 14 9d 2e  f9
1341 : 03 e8 d0 f7 ad 00 dd 30  9e
1349 : fb 20 04 14 a2 00 bd b4  62
1351 : 15 9d 2e 03 e8 d0 f7 20  56
1359 : 04 14 20 8e fb a2 01 20  5a
1361 : d1 fc b0 1f a0 34 84 01  80
1369 : a0 00 b1 ac a0 37 84 01  e3
1371 : 9d 2f 03 e8 20 db fc e0  1a
1379 : ff 90 e4 20 d1 fc b0 03  cb
1381 : a9 ff 2c a9 00 48 8d 2e  3f
1389 : 03 8e 2f 03 20 04 14 a2  b7
1391 : 01 68 d0 cb 18 a9 00 48  d3
1399 : ad 11 d0 09 10 8d 11 d0  77
13a1 : ad 1f 01 8d 00 dd 78 20  e1
13a9 : 15 fd 68 60 b9 2e 03 85  07
13b1 : 95 a2 00 2c 00 dd 50 fb  45
13b9 : 2c 00 dd 10 05 e8 d0 f8  2c
13c1 : f0 d2 a9 10 8d 00 dd aa  2c
13c9 : 66 95 6a 66 95 6a 4a 4a  cc
13d1 : 8d 00 dd 8a 66 95 6a 66  b0
13d9 : 95 6a 4a 4a 8d 00 dd 8a  e5
13e1 : 66 95 6a 66 95 6a 4a 4a  e4
13e9 : 8d 00 dd 8a 66 95 6a 66  c8
13f1 : 95 6a 4a 4a 8d 00 dd ea  bd
13f9 : ea ea a9 00 8d 00 dd c8  a5
1401 : d0 aa 60 a0 00 98 59 2e  d9
1409 : 03 c8 d0 fa 85 14 20 ad  d9
1411 : 13 88 a5 14 4c b0 13 4d  85
1419 : 2d 57 4d 2d 45 b7 01 a0  42
1421 : 00 84 11 a9 02 8d 00 18  9a
1429 : a9 04 2c 00 18 f0 fb a9  2c
1431 : 00 8d 00 18 a2 03 ca d0  0a
1439 : fd a2 0a ad 00 18 4a 6a  7e
1441 : 4a 66 85 0a 0a 66 85 ad  a6
1449 : 00 18 4a 6a 4a 66 85 0a  37
1451 : 0a 66 85 ad 00 18 4a 6a  64
1459 : 4a 66 85 0a 0a 66 85 ad  be
1461 : 00 18 8e 00 18 4a 6a 4a  23
1469 : 66 85 0a 0a 66 85 a5 85  8a
1471 : 91 30 45 11 85 11 c8 d0  33
1479 : aa 88 60 85 31 20 46 01  5f
1481 : b1 30 48 20 4a 01 68 91  d2
1489 : 30 a5 11 f0 ed 4c 43 e8  0e
1491 : 78 a9 0a 8d 00 18 a2 00  5d
1499 : 88 d0 fd ca d0 fa a9 0a  02
14a1 : 85 69 a9 00 85 30 a9 03  cc
14a9 : 20 a2 01 a9 04 20 a2 01  5e
14b1 : 4c 3d 04 a5 00 a2 01 86  78
14b9 : 00 29 02 f0 10 a6 98 86  92
14c1 : 32 20 2e 04 20 af 03 20  db
14c9 : 03 04 4c 69 f9 a2 08 20  24
14d1 : 35 03 a2 0a 20 35 03 a2  6f
14d9 : 08 20 f5 03 a2 0a 20 f5  b6
14e1 : 03 a5 8c 30 e8 4c 69 f9  6a
14e9 : 86 32 86 98 a5 8c 10 61  ff
14f1 : b5 83 d0 5d bd 31 04 20  fd
14f9 : a2 01 c8 a6 32 a9 80 95  c0
1501 : 83 a5 80 95 00 a5 81 95  88
1509 : 01 b1 30 f0 1c 20 21 f1  38
1511 : a6 82 f6 b5 d0 02 f6 bb  dd
1519 : a0 00 a5 80 91 30 c8 a5  3c
1521 : 81 91 30 a5 80 c5 22 f0  cc
1529 : 02 84 8c a6 32 a9 00 85  e1
1531 : 30 85 33 85 2e 85 36 85  94
1539 : 0c 85 50 a9 bb 85 34 bd  85
1541 : 31 04 85 2f 20 e9 f5 85  f0
1549 : 3a bd 32 04 20 a3 f7 a6  bb
1551 : 32 b5 83 f0 52 20 2e 04  44
1559 : ad 00 1c 29 10 d0 03 4c  5f
1561 : 81 f5 20 10 f5 a2 09 50  20
1569 : fe b8 ca d0 fa a9 ff 8d  a8
1571 : 03 1c ad 0c 1c 29 1f 09  09
1579 : c0 8d 0c 1c a9 ff a2 05  b6
1581 : 8d 01 1c b8 50 fe b8 ca  22
1589 : d0 fa a0 bb b1 0c 50 fe  31
1591 : b8 8d 01 1c c8 d0 f5 b1  22
1599 : 30 50 fe b8 8d 01 1c c8  ab
15a1 : d0 f5 50 fe 4c 00 fe 60  e1
15a9 : 86 32 2c 24 04 86 98 b5  1a
15b1 : 83 f0 f4 20 2e 04 20 0a  85
15b9 : f5 a0 bb b1 0c 50 fe b8  d4
15c1 : 4d 01 1c d0 19 c8 d0 f3  b3
15c9 : b1 30 50 fe b8 4d 01 1c  b8
15d1 : d0 0c c8 c0 fd d0 f1 a6  6d
15d9 : 32 a9 00 95 83 60 4c c5  8b
15e1 : f6 bd 31 04 85 31 bd 32  c0
15e9 : 04 85 0d 60 05 01 06 04  78
15f1 : 58 20 19 f1 a9 84 d5 a7  43
15f9 : f0 05 95 a7 20 42 d0 a9  71
1601 : 40 8d f9 02 a9 01 85 83  86
1609 : 20 07 d1 90 03 4c f8 cf  49
1611 : 20 3e de f6 b5 a9 00 85  9a
1619 : 8b 85 8d a9 80 85 8c a5  b1
1621 : 80 85 06 a9 e0 85 00 a5  a0
1629 : 00 30 fc f0 24 c9 01 f0  15
1631 : ea a5 18 85 06 a5 19 85  a2
1639 : 07 a2 00 a9 b0 20 7d d5  74
1641 : 20 99 d5 a9 e2 20 7d d5  a9
1649 : 20 99 d5 a5 8c d0 cc f0  c4
1651 : ce 4c 23 db 2a 30 31 2a  c7
1659 : 30 32 2a 30 33 2a 30 34  e1
1661 : 2a 30 35 2a 30 36 2a 30  f4
1669 : 37 2a 30 38 2a 30 39 2a  26
1671 : 31 30 2a 31 31 2a 31 32  f9
1679 : 2a 31 33 2a 31 34 2a 31  0e
1681 : 35 2a 31 36 2a 31 37 2a  3c
1689 : 31 38 2a 31 39 2a 32 30  95
1691 : 2a 32 31 2a 32 32 2a 32  28
1699 : 33 2a 32 34 2a 32 35 2a  52
16a1 : 32 36 2a 32 37 2a 32 38  bd
16a9 : 32 32 39 2a 33 30 00 00  35
16b1 : 00 cc 00 00 00 00 00 00  18
```

Listing. »Hypra-Copy«. Bitte mit dem MSE (Seite 136) eingeben.

Turbo Tape de Luxe

Machen Sie Ihrer Datasette Beine: Mit diesem Programm wird sie sogar etwas schneller als das Floppy-Laufwerk 1541. Damit wird die Datasette zur preiswerten Alternative.

Datasettenbesitzer müssen eigentlich gemütliche Leute sein. Bis zu zwanzig Minuten kann es nämlich dauern, wenn man ein umfangreiches Programm laden will. Viele Leute haben deswegen schon schnelle Lade- und Speicherroutinen geschrieben, die die Datasette um Faktoren zwischen 5 und 15 beschleunigen. Am bekanntesten und verbreitetsten ist das Programm »Turbo Tape«.

Das hier abgedruckte Programm »Turbo Tape de Luxe« (siehe Listing) zeichnet sich vor anderen Schnelladesystemen durch folgende Punkte aus:
- Es ist kompatibel zu »Turbo Tape«, kann also mit »Turbo Tape« gespeicherte Programme lesen. Daraus folgt, daß der Kassettenbetrieb um den Faktor 10 (wie bei »Turbo Tape«) beschleunigt wurde. Das ist immerhin sogar etwas schneller als das Floppy-Laufwerk 1541 (ohne Hypra-Load).
- Es arbeitet nicht mit neuen Basic-Befehlen, sondern mit einem Eingabemenü, so daß keine Konflikte mit Basic-Erweiterungen entstehen können.
- Das Basic-ROM kann während des Betriebs abgeschaltet werden, so daß auch Programme, die teilweise unter dem Basic-ROM liegen (lange Spiele und ähnliches), gespeichert werden können.
- Das Programm schützt sich selbst vor dem Überladen durch andere Programme.
- Es können beliebige RAM-Bereiche gespeichert werden.
- Programme können codiert und decodiert werden, damit kein Unbefugter Zugriff auf die Programme hat.
- Mit Hilfe des SMON kann das Programm in beliebige Speicherbereiche geschoben und sogar in ein EPROM gebrannt werden.

Nach dem Abtippen mit dem MSE und Speichern des Programms, kann es mit »SYS 49152« gestartet werden. Vorher sollten Sie jedoch NEW eingeben. Das Hauptmenü erscheint dann sofort und wartet auf eine Eingabe. Mit der Taste »Q« kann man dieses Menü wieder verlassen und dann später mit dem Befehl »£T« wieder ins Menü zurückkehren. Damit dies funktioniert, wird allerdings ein Basic-Vektor verbogen, der von vielen Basic-Erweiterungen benutzt wird und somit eine Zusammenarbeit mit »Turbo Tape de Luxe« völlig unmöglich macht. Soll dieser Vektor nicht verbogen werden, muß »Turbo Tape de Luxe« mit »SYS 49164« gestartet werden. An den Menüfunktionen hat sich nichts geändert, allerdings kann nach dem Verlassen mit »Q« nicht mehr mit »£T« ins Menü zurückgesprungen werden, sondern nur mit dem angegebenen SYS-Befehl.

Die Kommandos

Doch nun zur Erklärung der einzelnen Menü-Kommandos:

L - Load

Nach dem Drücken der Taste »L« wird vom Benutzer die Eingabe des Namens des zu ladenden Programmes verlangt. Soll das nächstbeste Programm geladen werden, darf der Name weggelassen werden. Ebenso ist eine Angabe mit Abkürzung (Joker) (»*«) erlaubt (Beispiel : »TUR*«). Da bei fast jeder Programmoption ein Filename angegeben werden muß, wird im einzelnen nicht mehr darauf hingewiesen.

Nachdem das Programm gefunden wurde, werden Name und Adresse des Programmes angezeigt und auf einen Druck auf die Commodore-Taste gewartet. Wird diese Taste nicht gedrückt, wird nach zirka acht Sekunden der Ladevorgang fortgesetzt.

P - Proload

Mit dieser Funktion können sowohl Basic-, wie auch Maschinensprache-Programme geladen und automatisch gestartet werden. Ist das eingeladene Programm ein Basic-Programm, wird der »RUN«-Befehl simuliert. Ist das geladene Programm hingegen ein Maschinen-Programm, wird es nach dem Laden an der Anfangsadresse gestartet. Dies funktioniert nicht bei allen Maschinen-Programmen!

V - Verify

Mit Verify kann ein abgespeichertes Programm mit dem im Speicher befindlichen verglichen werden.

S - Save

Speichert das im RAM vorhandene Basic-Programm.

A - Allsave

Speichert beliebige RAM-Bereiche ab. Die Start- und Endadresse muß in hexadezimaler Form angegeben werden. Beim Laden wird das Programm wieder an diese Adressen zurückgeladen.

O - Original

Speichert RAM-Bereiche so ab, daß sie später an andere Adressen geladen werden. Als erstes müssen Sie die Adressen angeben, an die das Programm später geladen werden soll. Danach geben Sie die Adressen an, an denen sich das Programm jetzt befindet. Zum Schluß folgt, wie bei allen Save-Befehlen, der Filename.

R - Renew

Holt ein durch »NEW« oder einen Reset gelöschtes Basic-Programm wieder zurück.

K - Kill

Zerstört »Turbo Tape de Luxe« und »biegt« die Vektoren wieder gerade. Zum Neuaufruf muß es erneut geladen werden.

M - Monitor

Löscht den Bildschirm und startet einen Maschinensprache-Monitor, sofern dieser im Speicher vorhanden ist. Damit individuelle Monitore verwendet werden können, muß die Einsprungadresse in $C52C/$C52D angepaßt werden. Ist kein Monitor vorhanden, kann das Programm abstürzen!

C - Code

Codiert einen Speicherbereich nach Adressenangabe. Das so codierte Programm kann gespeichert und später wieder geladen und decodiert werden.

D - Decode

Decodiert einen bestimmten Speicherbereich.

Wer sich einen individuellen Code erstellen möchte, damit außer ihm niemand Zugriff auf die Programme hat, muß folgende zwei Speicherstellenpaare ändern:

CODE1 : C591 und C59C
CODE2 : C594 und C59A

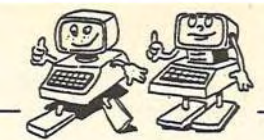
In den beiden Speicherstellen eines Paares muß immer dasselbe Byte stehen!

Hinweis zum Abtippen
Dieses Listing muß mit dem MSE eingegeben werden. Den MSE finden Sie in dieser Ausgabe auf Seite 136.

```
programm : t tape de luxe  c000 c9c0
--------------------------------------
c000 : ad 8d c0 8d 08 03 ad 8e  c2
c008 : c0 8d 09 03 a9 00 8d 20  43
c010 : d0 8d 21 d0 20 68 c2 20  9a
c018 : e4 ff f0 fb 48 20 76 c0  98
c020 : a9 9e 20 d2 ff 68 20 d2  e4
c028 : ff a2 10 dd 5d c6 f0 06  14
c030 : ca 10 f8 4c 0c c0 86 b4  14
c038 : 8a 0a 65 b4 aa bd 48 c0  f2
c040 : 48 bd 47 c0 48 60 2c a3  d0
c048 : c2 2c 96 c3 2c ab c3 2c  c6
c050 : b8 c3 2c bb c3 2c 25 c5  2a
c058 : 2c 2d c5 2c 3e c5 2c 44  5d
c060 : c5 2c b1 c5 2c b8 c5 2c  58
c068 : bf c5 2c ca c5 2c 0f c6  f6
c070 : 2c 00 00 2c 00 00 a0 11  c7
c078 : 2c a0 02 2c a0 0f 2c a0  6f
c080 : 16 a2 09 4c 0c e5 a2 14  56
c088 : a0 06 d0 f7 2c 8f c0 20  e1
c090 : 73 00 f0 04 c9 5c f0 03  09
c098 : 4c e7 a7 20 73 00 c9 54  cd
c0a0 : f0 03 4c 08 af 4c 0c c0  35
c0a8 : 49 ff 4c 0c e1 a2 00 bd  34
c0b0 : 6f c7 f0 06 20 a8 c0 e8  1c
c0b8 : d0 f5 bd 6f c8 f0 06 20  4c
c0c0 : a8 c0 e8 d0 f5 bd 6f c9  bb
c0c8 : f0 06 20 a8 c0 e8 d0 f5  5b
c0d0 : 60 8a 48 20 86 c0 a9 9e  de
c0d8 : 20 d2 ff 68 aa bd 73 c6  62
c0e0 : f0 06 20 a8 c0 e8 d0 f5  73
c0e8 : 60 a2 38 20 d1 c0 a9 02  79
c0f0 : 8d 8b 02 a5 cb c9 3c f0  56
c0f8 : 02 d0 f3 4c 0c c0 a2 70  1b
c100 : 2c a2 8c a9 c4 48 20 d1  88
c108 : c0 68 aa 4c 70 c1 a2 8c  e9
c110 : a9 a8 d0 f1 a9 08 85 a1  b4
c118 : 20 e0 e4 4c e9 c0 a2 e0  5c
c120 : 20 d1 c0 4c ee c0 85 02  f1
c128 : 4a 4a 4a 4a 20 33 c1 a5  61
c130 : 02 29 0f c9 0a 30 06 69  d1
c138 : 36 20 ca f1 60 69 30 d0  23
c140 : f8 20 79 c0 a5 40 20 26  e8
c148 : c1 a5 3f 20 26 c1 a9 2d  21
c150 : 20 ca f1 a5 42 20 26 c1  48
c158 : a5 41 4c 26 c1 20 7f c0  12
c160 : a2 00 bd 41 03 20 ca f1  da
c168 : e8 e0 10 d0 f5 4c 41 c1  29
c170 : 20 d1 c0 4c 14 c1 00 00  82
c178 : 00 00 00 00 00 2c c0 02  e1
c180 : ad 7e c1 85 bb ad 7f c1  38
c188 : 85 bc 60 20 79 c0 a2 00  b0
c190 : a9 20 20 ca f1 e8 e0 09  a7
c198 : d0 f8 60 20 80 c1 a9 a4  06
c1a0 : 20 ca f1 a0 00 84 b7 20  f9
c1a8 : 3e f1 f0 fb c9 93 f0 f7  87
c1b0 : c9 13 f0 f3 c9 11 f0 ef  86
c1b8 : c9 91 f0 eb c9 9d f0 e7  20
c1c0 : c9 1d f0 e3 c9 14 f0 1e  0e
c1c8 : c9 0d f0 2c a4 b7 91 bb  9f
c1d0 : 48 a9 9d 20 ca f1 68 20  76
c1d8 : ca f1 c4 ba f0 c7 a9 a4  60
c1e0 : 20 ca f1 c8 d0 bf a2 00  90
c1e8 : bd 6d c6 f0 06 20 ca f1  9c
c1f0 : e8 d0 f5 c6 b7 4c a7 c1  96
c1f8 : a4 b7 c4 ba f0 0a a9 9d  41
c200 : 20 ca f1 a9 20 20 ca f1  49
c208 : 60 20 80 c1 a0 00 a9 20  c2
c210 : 91 bb c8 c0 10 d0 f9 60  f9
c218 : 20 79 c0 a9 08 85 ba d0  93
c220 : 07 20 7f c0 a9 0f 85 ba  ce
c228 : 20 09 c2 4c 9b c1 20 3f  ce
c230 : c2 0a 0a 0a 0a 85 02 c8  22
c238 : 20 3f c2 18 65 02 60 b1  f7
c240 : bb c9 3a 30 04 38 e9 37  8d
c248 : 60 38 e9 30 60 a0 00 20  90
c250 : 2e c2 85 42 c8 20 2e c2  55
c258 : 85 41 c8 c8 20 2e c2 85  53
c260 : 44 c8 20 2e c2 85 43 60  fc
c268 : 20 ad c0 20 7c c0 a9 01  09
c270 : 2c 01 00 f0 05 a9 31 4c  36
c278 : ca f1 a9 30 d0 f9 a9 00  2f
c280 : a8 aa 4c ba ff a2 04 b5  78
c288 : 40 95 ab ca d0 f9 60 20  76
c290 : 18 c2 4c 4d c2 20 8f c2  b7
c298 : 20 85 c2 20 8b c1 20 8f  96
c2a0 : c2 4c 21 c2 a2 05 86 ab  ed
c2a8 : 20 9e c2 20 85 c2 20 7e  b8
c2b0 : c2 a2 1c 20 d1 c0 20 d0  14
c2b8 : f8 20 2e f8 d0 f5 20 83  af
c2c0 : c3 20 63 c3 a5 b9 18 69  40
c2c8 : 01 ca 20 35 c3 a2 08 b9  c2
c2d0 : ac 00 20 35 c3 a2 06 c8  26
c2d8 : c0 05 d0 f3 a0 00 a2 04  6a
c2e0 : b1 bb c4 b7 90 03 a9 20  9f
c2e8 : ca 20 35 c3 a2 05 c8 c0  7f
c2f0 : bb d0 ed a9 02 85 ab 20  ff
c2f8 : 63 c3 98 20 35 c3 84 d7  9a
c300 : a2 07 b1 41 20 35 c3 a2  ba
c308 : 03 e6 41 d0 04 e6 42 ca  ff
c310 : ca a5 41 c5 43 a5 42 e5  ec
c318 : 44 90 e7 a5 d7 20 35 c3  2e
c320 : a2 07 88 d0 f6 c8 84 c0  cb
c328 : 58 18 a9 00 8d a0 02 20  1d
c330 : 93 fc 4c e9 c0 85 bd 45  4b
c338 : d7 85 d7 a9 08 85 a3 06  44
c340 : bd a5 01 29 f7 20 56 c3  97
c348 : a2 11 09 08 20 56 c3 a2  bf
c350 : 0e c6 a3 d0 ea 60 ca d0  43
c358 : fd 90 05 a2 0b ca d0 fd  79
c360 : 85 01 60 a0 00 a9 02 20  28
c368 : 35 c3 a2 07 88 c0 09 d0  5d
c370 : f4 a2 05 c6 ab d0 ee 98  fe
c378 : 20 35 c3 a2 07 88 d0 f7  60
c380 : ca ca 60 a0 00 84 c0 ad  5e
c388 : 11 d0 29 ef 8d 11 d0 ca  84
c390 : d0 fd 88 d0 fa 78 60 a2  d5
c398 : 05 86 ab a2 04 b5 2a 95  e1
c3a0 : ab 95 40 ca d0 f7 20 21  0f
c3a8 : c2 4c ae c2 a2 05 86 ab  58
c3b0 : 20 95 c2 4c ae c2 4c ae  64
c3b8 : c2 a2 00 2c a2 01 86 0a  b1
c3c0 : 86 93 20 21 c2 20 ee c3  ac
c3c8 : a5 c3 85 2d a5 c4 85 2e  49
c3d0 : a5 0a f0 0d 20 b7 ff 29  6a
c3d8 : bf d0 03 4c 0e c1 4c 01  6c
c3e0 : c1 20 b7 ff 29 bf f0 03  f9
c3e8 : 4c fe c0 4c e9 c0 20 76  7f
c3f0 : c4 a5 ab c9 02 f0 04 c9  f6
c3f8 : 01 d0 f3 ad 3c 03 85 c3  8d
c400 : 85 3f ad 3d 03 85 c4 85  b2
c408 : 40 ad 3e 03 85 41 ad 3f  a6
c410 : 03 85 42 20 5d c1 2c 00  ff
c418 : c0 2c ff cf ad 14 c4 cd  12
c420 : 3e 03 ad 15 c4 ed 3f 03  ad
c428 : b0 16 ad 17 c4 cd 3c 03  e3
c430 : ad 18 c4 ed 3d 03 90 08  16
c438 : a2 54 20 d1 c0 4c 14 c1  89
c440 : a9 10 85 a1 20 e0 e4 a5  6f
c448 : cb c9 3f d0 03 4c 0c c0  26
c450 : a4 b7 f0 0f 88 b1 bb c9  86
c458 : 2a f0 05 d9 41 03 d0 8e  03
c460 : 98 d0 f1 84 90 [illegible] 8c c4  33
c468 : a5 bd 45 d7 05 90 f0 04  d9
c470 : a5 ff 85 90 18 60 20 c6  1b
c478 : c4 c9 00 f0 f9 85 ab 20  fa
c480 : f4 c4 91 b2 c8 c0 c0 d0  c8
c488 : f6 f0 2e ea 20 c6 c4 20  6b
c490 : f4 c4 c4 93 d0 02 91 c3  75
c498 : d1 c3 f0 02 86 90 45 d7  79
c4a0 : 85 d7 e6 c3 d0 02 e6 c4  85
c4a8 : a5 c3 c5 41 a5 c4 e5 42  65
c4b0 : 90 dd 20 f4 c4 20 83 c3  b8
c4b8 : c8 84 c0 18 58 a9 00 8d  e3
c4c0 : a0 02 4c 93 fc ea 20 1b  c5
c4c8 : c5 20 83 c3 84 d7 a9 07  b2
c4d0 : 8d 06 dd a2 01 20 04 c5  d9
c4d8 : 26 bd a5 bd c9 02 d0 f5  da
c4e0 : a0 09 20 f4 c4 c9 02 f0  30
c4e8 : f9 c4 bd d0 e8 20 f4 c4  ba
c4f0 : 88 d0 f6 60 a9 08 85 a3  e2
c4f8 : 20 04 c5 26 bd c6 a3 d0  93
c500 : f7 a5 bd 60 a9 10 2c 0d  2b
c508 : dc f0 fb ad 0d dd 8e 07  19
c510 : dd 48 a9 19 8d 0f dd 68  38
c518 : 4a 4a 60 a2 00 20 d1 c0  be
c520 : 20 2e f8 d0 fb 60 a9 93  40
c528 : 20 ca f1 4c 00 90 a2 36  2f
c530 : a9 30 86 01 48 20 7c c0  ac
c538 : 68 20 ca f1 4c 17 c0 a2  67
c540 : 37 a9 31 d0 ed a5 2b a4  b4
c548 : 2c 85 22 84 23 a0 03 c8  25
c550 : b1 22 d0 fb c8 98 18 65  42
c558 : 22 a0 00 91 2b a5 23 69  3c
c560 : 00 c8 91 2b 88 a2 03 e6  06
c568 : 22 d0 02 e6 23 b1 22 d0  3a
c570 : f4 ca d0 f3 a5 22 69 02  91
c578 : 85 2d a5 23 69 00 85 2e  6b
c580 : 4c 1e c1 20 8f c2 a0 00  e1
c588 : b1 41 a6 ba e0 70 f0 08  40
c590 : 49 ff 18 69 20 4c 9d c5  72
c598 : 38 e9 20 49 ff 91 41 e6  55
c5a0 : 41 d0 02 e6 42 a5 43 c5  91
c5a8 : 41 a5 44 e5 42 b0 d7 4c  2b
c5b0 : 1e c1 a9 00 85 ba 4c 83  80
c5b8 : c5 a9 70 85 ba 4c 83 c5  c6
c5c0 : a9 93 20 ca f1 a2 f6 9a  d9
c5c8 : 6c 00 03 ad 09 03 cd 8e  a8
c5d0 : c0 d0 0a a9 e4 8d 08 03  91
c5d8 : a9 a7 8d 09 03 2c 00 c0  ec
c5e0 : 2c bf c5 ad de c5 85 41  c7
c5e8 : ad df c5 85 42 ad e1 c5  4b
c5f0 : 85 43 ad e2 c5 85 44 a9  cb
c5f8 : 00 a0 00 91 41 e6 41 d0  6d
c600 : 02 e6 42 a5 43 c5 41 a5  6d
c608 : 44 e5 42 b0 ea 4c c0 c5  85
c610 : a2 00 2c a2 01 86 0a 86  8b
c618 : 93 20 21 c2 20 ee c3 a5  30
c620 : c3 85 2d a5 c4 85 2e a5  22
c628 : 0a f0 0d 20 b7 ff 29 bf  91
c630 : d0 03 4c 0e c1 4c 01 c1  5d
c638 : 20 b7 ff 29 bf f0 03 4c  81
c640 : fe c0 a9 93 20 ca f1 a5  e6
c648 : 3f a6 40 c5 2b d0 04 e4  b6
c650 : 2c f0 03 6c 3f 00 20 59  6a
c658 : a6 4c ae a7 ff 41 53 4f  bb
c660 : 4c 56 4d 2d 2b 52 43 44  ab
c668 : 51 4b 50 00 00 9d 20 9d  1c
c670 : 9d a4 00 df df df df df  98
c678 : af ad ba ac ac df af b3  32
c680 : be a6 df b0 b1 df ab be  e5
c688 : af ba df df df df 00 af  e4
c690 : ad ba ac ac df ad ba bc  2b
c698 : b0 ad bb df d9 df af b3  cc
c6a0 : be a6 df b0 b1 df ab be  05
c6a8 : af ba 00 df df df df df  ed
c6b0 : df df df af ad ba ac ac  29
c6b8 : df ac af be bc ba df df  92
c6c0 : df df df df df df 00 df  3f
c6c8 : df df df df b3 b0 be bb  be
c6d0 : df b2 ba b2 b0 ad a6 df  e0
c6d8 : ba ad ad b0 ad df df df  03
c6e0 : df df 00 df df df df df  e7
c6e8 : df df df df df b3 b0 be  86
c6f0 : bb df df df df df df df  cb
c6f8 : df df df df df df 00 df  77
c700 : df df df df df df df a9  92
c708 : ba ad b6 b9 a6 df df df  26
c710 : df df df df df df df df  0f
c718 : df df 00 e2 e2 e2 e2 e2  da
c720 : e2 e2 e2 e2 e2 e2 e2 e2  1f
c728 : e2 df b0 b4 df df df df  f9
c730 : df df df df df df 00 e2  b5
c738 : e2 e2 e2 e2 e2 e2 e2 e2  37
c740 : e2 e2 e2 e2 e2 df ba ad  1c
c748 : ad b0 ad df df df df df  f1
c750 : df df 00 df df df df bb  0f
c758 : b0 b1 ba df d1 d1 d1 df  3e
c760 : af ad ba ac ac df ac af  06
c768 : be bc ba df df df 00 6c  05
c770 : 60 ed df df df df ab df  26
c778 : aa df ad df bd df b0 df  d6
c780 : df ab df be df af df ba  75
c788 : df df bb df ba df df b3  d3
c790 : df aa df a7 df ba df df  c4
c798 : df df df df df df df df  97
c7a0 : df df df df df df df df  9f
c7a8 : df df df df df df df df  a7
c7b0 : df df df df df df df df  af
c7b8 : df df df df df df df df  b7
c7c0 : df df 61 df df df df df  1f
c7c8 : df a8 ad b6 ab ab ba b1  a4
c7d0 : df bd a6 df b5 b0 b7 be  71
c7d8 : b1 b1 ba ac df b8 b0 b3  94
c7e0 : b0 b2 bd ba b4 df df df  39
c7e8 : df df df df df df df df  e7
c7f0 : df df df b1 ba aa b2 aa  0e
c7f8 : ba b1 ac ab ba ad d0 bd  03
c800 : ad be aa b1 ac bc b7 a8  ce
c808 : ba b6 b8 df df df df df  84
c810 : df df df f2 f2 f2 df ed  57
c818 : df df be bb ad ba ac ac  cb
c820 : ba df df 6d df df ed bd  9f
c828 : ad df bd ba 6d df df ed  bd
c830 : df df df df df b9 b6 b3  01
c838 : ba b1 be b2 ba df df df  bb
c840 : df df f2 df 4f 3c 3c 3c  08
c848 : 3c 3c 3c 3c 3c 3c 51 df  e4
c850 : df 4f 3c 4d 3c 51 df df  1d
c858 : 4f 3c 3c 3c 3c 3c 3c 3c  6b
c860 : 3c 3c 3c 3c 3c 3c 3c 3c  60
c868 : 3c 51 df df 3d df df df  53
c870 : df df df df df df 3d df  e4
c878 : df 3d df 3d df 3d df df  bc
c880 : 3d df df df df df df df  dd
c888 : df df df df df df df df  87
c890 : df 3d df df 52 3c 3c 3c  72
c898 : 3c 3c 3c 3c 3c 3c 42 df  f7
c8a0 : df 52 3c 4e 3c 42 df df  96
c8a8 : 52 3c 3c 3c 3c 3c 3c 3c  be
c8b0 : 3c 3c 3c 3c 3c 3c 3c 3c  b0
c8b8 : 3c 42 f2 f2 f2 f2 c3 b3  6d
c8c0 : c1 b0 be bb df c5 df c3  33
c8c8 : a9 c1 ba ad b6 b9 a6 df  4a
c8d0 : df c5 df c3 af c1 ad b0  23
c8d8 : b3 b0 be bb df c5 df c3  3d
```

Listing zu »Turbo Tape de Luxe«. Beachten Sie bitte die Eingabehinweise auf Seite 136.





Q - Quit

Verläßt das »Turbo Tape de Luxe«-Menü und springt zurück in Basic.

± - Basic-ROM ein-/ausschalten

Im »Turbo Tape de Luxe«-Menü kann das Basic-ROM ($A000-$BFFF) abgeschaltet werden, um Programme, Grafikbilder und anderes im RAM unter dem ROM aus diesem Bereich speichern zu können. Der derzeitige Zustand des Basic-ROMs wird in dem Kästchen »BR« angezeigt.

Die abgedruckte Version von »Turbo Tape de Luxe« liegt im Speicherbereich von $C000 bis $C9B0. Sie kann mit dem SMON und seinem »C«-Befehl in beliebige Speicherbereiche geschoben werden, da alle verwendeten Sprungtabellen als BIT-Befehle getarnt sind. Genaueres zur Verwendung des »C«-Befehls beim SMON entnehmen Sie bitte der Anleitung zum SMON selbst (64'er, Ausgabe 11/84 bis 2/85).

Wer will, kann »Turbo Tape de Luxe« nach dem Verschieben sogar in ein EPROM brennen.

Sollte sich »Turbo Tape de Luxe« einmal mit »Load Memory Error« melden, dann haben Sie versucht, ein Programm zu laden, das »Turbo Tape de Luxe« teilweise überschrieben und gelöscht und somit einen sauberen Systemabsturz heraufbeschworen hätte. Diese Sicherheitsabfrage funktioniert auch in etwaigen verschobenen Versionen. (J. Golombek/bs)

```
c8e0 : ae c1 aa b6 ab df c3 ac   12
c8e8 : c1 be a9 ba df c5 df c3   fd
c8f0 : be c1 b3 b3 ac be a9 ba   cf
c8f8 : df c5 df c3 b0 c1 ad b6   67
c900 : b8 b1 be b3 df c5 df c3   ea
c908 : b4 c1 b6 b3 b3 df c3 bc   84
c910 : c1 b0 bb ba df c5 df c3   a3
c918 : bb c1 ba bc b0 bb ba df   8d
c920 : df c5 df c3 b2 c1 b0 b1   b1
c928 : b6 ab b0 ad df c5 df c3   c9
c930 : ad c1 ba b1 ba a8 f2 df   1f
c938 : df df df df ed df df df   18
c940 : df df ac a6 ac ab ba b2   87
c948 : df b0 ad bb ba ad ac d0   cf
c950 : ba ad ad b0 ad ac df df   e2
c958 : df df df f2 df df df df   b9
c960 : df 4f 3c 3c 3c 3c 3c 3c   8d
c968 : 3c 3c 3c 3c 3c 3c 3c 3c   68
c970 : 3c 3c 3c 3c 3c 3c 3c 3c   70
c978 : 3c 3c 3c 3c 3c 3c 51 ee   32
c980 : 62 3d 6e f2 df df df df   b7
c988 : df 3d f2 df df df df df   fb
c990 : 52 3c 3c 3c 3c 3c 3c 3c   a6
c998 : 3c 3c 3c 3c 3c 3c 3c 3c   98
c9a0 : 3c 3c 3c 3c 3c 3c 3c 3c   a0
c9a8 : 3c 3c 3c 3c 3c 42 00 00   6f
c9b0 : 00 ff 00 ff 00 ff 00 ff   b0
c9b8 : 00 ff 00 ff 00 ff 00 ff   b8
```

Listing »Turbo Tape de Luxe« (Schluß).
Beachten Sie bitte die Eingabehinweise auf Seite 136.

Hypra-Load

Kaffeepause! Dieses Wort kommt wohl jedem Commodore-64-Besitzer in den Sinn, der ein längeres Programm von der Diskette laden möchte. Es geht aber fünfmal schneller. Alles, was Sie benötigen, ist das Programm »Hypra-Load«.

Daß die Floppy 1541 nicht gerade die schnellste ist, ist wohl allgemein bekannt. Zumindest ein Vorgang läßt sich mit relativ geringem Aufwand stark beschleunigen: Das Laden von Programmen.

Das Funktionsprinzip von Hypra-Load sieht folgendermaßen aus. Im Computer wie auch im Floppy-Laufwerk werden neue Laderoutinen installiert, die neue Busroutinen enthalten. Mit diesen neuen Busroutinen wird die Datenübertragung um den Faktor 8 bis 10 beschleunigt, so daß die Ladegeschwindigkeit auf das softwaremäßig erreichbare Maximum gebracht wird.

Eine Software-Lösung hat allerdings zwei entscheidende Nachteile: Einerseits muß vor dem schnellen Laden die neue LOAD-Routine in den Speicher gebracht werden, andererseits funktionieren manche Originalprogramme mit Kopierschutz nicht mehr. Diese Nachteile wurden bei der hier vorgestellten Version so gut wie möglich umgangen.

»Hypra-Load 2.1« ist eine besonders komfortable Hypra-Load-Version mit integriertem Auto-Starter und Auto-Lader. Sie wurde von den Autoren des original »Hypra-Load« (aus der 64'er-Ausgabe 10/84) völlig neu geschrieben.

Um Hypra-Load beim späteren Betrieb möglichst komfortabel zu halten, ist die Eintipparbeit etwas komplizierter. Sie benötigen als allererstes eine leere Diskette, die Sie am besten formatieren. Nun tippen Sie das Listing 1 mit dem MSE ab und speichern es genau auf dieser leeren Diskette. Sie sollten keinesfalls probieren, auf eine Diskette zu speichern, auf der schon Programme stehen, weil diese beim nächsten Schritt unrettbar verloren gehen!

Programme schneller laden

Nun tippen Sie das kurze Listing 2 ab. Legen Sie die Diskette, auf der sich das gespeicherte MSE-Listing befindet, ein und starten Sie das Basic-Programm. Nach wenigen Sekunden meldet sich der Computer mit READY. Nun ist Ihre Hypra-Load-Version fertig. Um sie auf andere Disketten zu überspielen, können Sie jedes beliebige Kopierprogramm für Einzelfiles verwenden. Die Disketten, auf die Hypra-Load dann kopiert werden soll, dürfen ruhig schon Programme enthalten, es müssen aber insgesamt fünf Blöcke auf der Diskette frei sein, damit noch Platz für Hypra-Load ist.

Wie arbeitet man nun mit Hypra-Load? Es gibt zwei Betriebsarten: In der normalen Betriebsart lädt man Hypra-Load mit LOAD":*",8,1. Daraus folgt, daß es das erste Programm auf der Diskette sein muß! Hypra-Load installiert sich dann vollautomatisch, ohne daß RUN eingetippt werden muß, und meldet sich mit den Namen der Autoren. Ab sofort wird von Diskette fünfmal schneller geladen. Zusätzlich sind als Standardwerte für den LOAD-Befehl »,8,1« schon fest vorgegeben. »LOAD"name"« lädt also immer von Diskette.

Bei einem Reset schaltet sich Hypra-Load ab, ebenso wenn wichtige Speicherbereiche überschrieben werden und somit ein Systemabsturz provoziert wird.

Viel interessanter und komfortabler, Hypra-Load zu nutzen, ist allerdings die zweite Methode. Wenn Hypra-Load sich zusammen mit anderen Programmen auf einer Diskette befin-

Hinweis zum Abtippen
Dieses Listing muß mit dem MSE eingegeben werden. Den MSE finden Sie in dieser Ausgabe auf Seite 136.

```
programm : ↑ hypra-load    1000 14bd
-------------------------------------
1000 : a9 08 85 af a9 00 85 ae  13
1008 : a9 6a 8d 2c 03 a9 0b 8d  94
1010 : 2d 03 a9 ed 8d 28 03 a9  60
1018 : f6 8d 29 03 60 86 ae 00  75
1020 : 00 00 00 4c 48 b2 00 31  26
1028 : ea 66 fe 47 fe 4a f3 91  23
1030 : f2 0e f2 50 f2 33 f3 57  37
1038 : f1 ca f1 ed 02 a9 bb a2  6a
1040 : ee 85 a7 86 a8 a9 00 a2  c9
1048 : 03 85 a9 86 aa a9 08 20  a1
1050 : 0c ed a9 6f 20 b9 ed a9  86
1058 : 4d 20 dd ed a9 2d 20 dd  2b
1060 : ed a9 57 20 dd ed a0 00  cb
1068 : a5 a9 20 dd ed a5 aa 20  9d
1070 : dd ed a9 1d 20 dd ed b1  5e
1078 : a7 20 dd ed c8 c0 1d 90  8c
1080 : f6 20 fe ed 18 a5 a7 69  24
1088 : 1d 85 a7 90 03 e6 a8 18  9e
1090 : a5 a9 69 1d 85 a9 90 02  f4
1098 : e6 aa a5 a9 c9 d0 d0 ad  34
10a0 : a9 08 20 0c ed a9 6f 20  01
10a8 : b9 ed a9 4d 20 dd ed a9  68
10b0 : 2d 20 dd ed a9 45 20 dd  23
10b8 : ed a9 01 20 dd ed a9 04  ba
10c0 : 20 dd ed 20 fb ed 20 b3  65
10c8 : ee 20 b3 ee 78 a9 0b 8d  ad
10d0 : 11 d0 a9 00 85 bb a9 fe  8f
10d8 : 85 bc 20 39 f8 c9 ff f0  aa
10e0 : 61 20 39 f8 85 b7 20 39  c8
10e8 : f8 a8 a5 b7 d0 03 88 84  e5
10f0 : bc a5 bb d0 0a c6 bc c6  df
10f8 : bc 20 39 f8 20 39 f8 a0  23
1100 : 00 20 39 f8 91 ae c8 c4  b9
1108 : bc d0 f6 18 a5 bc 65 ae  20
1110 : 85 ae 90 02 e6 af e6 bb  50
1118 : a5 b7 d0 ba c6 bb d0 04  ba
1120 : e6 bc e6 bc a5 bc c9 fe  1b
1128 : f0 08 20 39 f8 e6 bc 18  35
1130 : 90 f2 a9 00 2c a9 1d 85  33
1138 : 90 4c 85 f8 a5 90 c9 01  77
1140 : 48 08 98 a6 ae a4 af 4c  ef
1148 : 61 00 a9 23 8d 00 dd 2c  21
1150 : 00 dd 50 fb a9 03 8d 00  bb
1158 : dd a2 07 ca d0 fd ad 00  55
1160 : dd 2a 2a 66 b0 6a 66 b0  03
1168 : ea ea ad 00 dd 2a 2a 66  d7
1170 : b0 6a 66 b0 ea ea ad 00  c2
1178 : dd 2a 2a 66 b0 6a 66 b0  1b
1180 : ea ea ad 00 dd 2a 2a 66  ef
1188 : b0 6a 66 b0 a5 b0 49 ff  22
1190 : a2 03 8e 00 dd 60 68 a8  2b
1198 : 68 aa 48 98 48 e0 e0 90  ab
11a0 : 0d a9 00 a8 59 00 a0 c8  41
11a8 : d0 fa c9 80 d0 10 a9 00  ac
11b0 : a8 59 bb ee c8 d0 fa c9  64
11b8 : 70 d0 03 a0 35 2c a0 37  0b
11c0 : a9 85 8d 61 00 a9 01 8d  28
11c8 : 62 00 a9 28 8d 63 00 a9  e1
11d0 : 68 8d 64 00 a9 60 8d 65  b7
11d8 : 00 a9 1b 8d 11 d0 4c 2b  44
11e0 : f8 a5 00 29 06 c9 02 f0  69
11e8 : 03 4c 9e fd ea a9 05 85  94
11f0 : 09 a2 5a 86 4b a2 00 a9  cf
11f8 : 52 85 24 20 56 f5 50 fe  6e
1200 : b8 ad 01 1c c5 24 f0 09  a6
1208 : c6 4b d0 ef a9 0a 4c 69  95
1210 : f9 50 fe b8 ad 01 1c 95  86
1218 : 25 e8 e0 07 d0 f3 20 97  27
1220 : f4 a5 16 45 17 45 18 45  9c
1228 : 19 45 1a f0 07 c6 09 d0  f5
1230 : c0 4c 1e f4 a5 18 c5 06  7b
1238 : f0 03 4c 0b f4 85 22 a9  76
1240 : 06 85 31 4c eb 03 a5 12  70
1248 : a6 13 85 16 86 17 a5 06  60
1250 : 85 18 a5 07 85 19 a9 00  f3
1258 : 45 16 45 17 45 18 45 19  39
1260 : 85 1a 20 34 f9 a2 5a 20  df
1268 : 56 f5 a0 00 50 fe b8 ad  1c
1270 : 01 1c d9 24 00 f0 06 ca  af
1278 : d0 ed 4c 51 f5 c8 c0 08  35
1280 : d0 ea 20 56 f5 50 fe b8  e7
1288 : ad 01 1c 91 30 c8 d0 f5  68
1290 : a0 ba 50 fe b8 ad 01 1c  b6
1298 : 99 00 01 c8 d0 f4 20 e0  82
12a0 : f8 a5 38 c5 47 f0 03 4c  d2
12a8 : f6 f4 20 e9 f5 c5 3a f0  b6
12b0 : 03 4c 02 f5 a0 00 a9 55  74
12b8 : 20 74 04 b9 00 06 20 74  e4
12c0 : 04 c8 d0 f7 ad 00 1c 49  39
12c8 : 08 8d 00 1c ad 00 06 d0  af
12d0 : 03 4c 9e fd c5 18 d0 f9  b5
12d8 : 85 06 ad 01 06 85 07 4c  2d
12e0 : 65 03 78 a9 08 8d 00 18  37
12e8 : a5 18 8d 00 06 85 06 a5  ed
12f0 : 19 8d 01 06 85 07 a9 04  10
12f8 : 85 78 a9 e2 20 6b 04 c9  81
1300 : 02 90 33 a0 00 84 78 a4  7b
1308 : 78 b9 db fe f0 12 58 20  75
1310 : 76 d6 78 a9 e2 20 [illegible] 04  29
1318 : c9 02 90 1a e6 78 d0 e7  8f
1320 : a9 c0 20 6b 04 a9 e2 20  f8
1328 : 6b 04 c9 02 90 08 a9 ff  38
1330 : 20 74 04 4c 22 eb ad 00  4d
1338 : 06 f0 f8 c5 18 f0 c4 ad  25
1340 : 00 06 85 06 ad 01 06 85  6b
1348 : 07 4c 15 04 85 00 58 a5  40
1350 : 00 30 fc 78 60 85 77 a2  0c
1358 : 01 8a 2c 00 18 f0 fb a9  f6
1360 : 00 8d 00 18 8a 2c 00 18  64
1368 : d0 fb a2 00 8a 66 77 2a  ed
1370 : 2a 66 77 2a 2a 8d 00 18  30
1378 : 8a 66 77 2a 2a 66 77 2a  61
1380 : 2a 8d 00 18 8a 66 77 2a  82
1388 : 2a 66 77 2a 2a 8d 00 18  48
1390 : 8a 66 77 2a 2a 66 77 2a  79
1398 : 2a 8d 00 18 a2 02 ca d0  93
13a0 : fd a9 08 8d 00 18 60 a0  a9
13a8 : 00 84 61 a9 a0 85 62 b1  9b
13b0 : 61 91 61 e6 61 d0 f8 e6  5d
13b8 : 62 f0 0c a5 62 c9 c0 d0  63
13c0 : ee a9 e0 85 62 d0 e8 a9  0f
13c8 : e5 8d d6 fd a9 4c 8d f9  10
13d0 : f4 a9 2c 8d fa f4 a9 f7  43
13d8 : 8d fb f4 a2 00 bd 00 08  f2
13e0 : 9d 2c f7 e8 d0 f7 bd 00  72
13e8 : 09 9d 2c f8 e8 e0 a4 d0  b4
13f0 : f5 a2 00 bd a4 09 9d bb  6f
13f8 : ee e8 d0 f7 bd a4 0a 9d  f2
1400 : bb ef e8 e0 c6 d0 f5 a9  27
1408 : 08 8d da e1 a9 01 8d dc  5c
1410 : e1 a9 35 85 01 a0 00 b1  3c
1418 : bb c9 5e f0 59 20 53 e4  1b
1420 : 20 bf e3 18 20 15 fd a9  12
1428 : f7 a0 0b 20 1e ab 20 44  7f
1430 : a6 4c 9d e3 0d 48 59 50  f9
1438 : 52 41 2d 4c 4f 41 44 20  50
1440 : 56 32 2e 31 20 28 43 29  04
1448 : 20 31 39 38 35 20 54 52  a1
1450 : 49 42 41 52 0d 42 4f 52  1a
1458 : 49 53 20 53 43 48 4e 45  f8
1460 : 49 44 45 52 2b 4b 41 52  1e
1468 : 53 54 45 4e 20 53 43 48  3b
1470 : 52 41 4d 4d 0d 00 18 a9  e4
1478 : 02 65 bb 85 bb 90 02 e6  e2
1480 : bc c6 b7 c6 b7 ea a2 00  c3
1488 : bd 59 0c 9d c0 02 e8 e0  2a
1490 : 25 d0 f5 4c c0 02 20 53  67
1498 : e4 20 bf e3 18 20 15 fd  cb
14a0 : 20 44 a6 20 d5 ff a9 37  02
14a8 : 85 01 a5 ae 85 2d a5 af  a5
14b0 : 85 2e 20 63 a6 20 8e a6  b4
14b8 : 4c ae a7 00 31 00 00 00  58
```

Listing 1. »Hypra-Load 2.1«. Bitte beachten Sie die Hinweise im Text

```
10 OPEN 1,8,15
20 OPEN 2,8,2,"#"
30 PRINT#1,"U1 2 0 17 0"
40 PRINT#1,"B-P 2 2"
50 PRINT#2,CHR$(237);CHR$(2);
60 PRINT#1,"U2 2 0 17 0"
70 CLOSE 2,1

READY.
```

Listing 2. Hilfsprogramm zur Erstellung von »Hypra-Load 2.1«. Bitte beachten Sie die Hinweise im Text!

det, genügt der Befehl »LOAD"↑*name",8,1«, um mehrere Vorgänge gleichzeitig in Gang zu setzen: Hypra-Load wird geladen und gestartet, danach wird vollautomatisch das Programm mit dem Namen »name« schnell nachgeladen und ebenfalls gestartet. Voraussetzung dabei ist, daß die Hypra-Load-Version auf der Diskette genau mit dem im MSE-Listing angegebenen Namen (»↑HYPRA-LOAD«) gespeichert ist.

Einige Anmerkungen zum Auto-Lader: Nach der Installation von Hypra-Load wird überprüft, ob der aktuelle Filename mit einem "↑" (Pfeil nach oben) beginnt. Dann werden die ersten beiden Buchstaben vom Filenamen abgetrennt und das gewünschte Programm nachgeladen. Nach dem Laden desaktiviert sich Hypra-Load sofort und startet das geladene Programm durch Simulation des RUN-Befehls. Mehrteilige Programme laufen also ab dem zweiten Teil in der normalen Ladegeschwindigkeit ab. Für die meisten Anwendungsfälle ist diese Lösung allerdings vollkommen befriedigend.

Diese Hypra-Load-Version läuft unserer Erfahrung nach sowohl auf C 64- wie auch SX 64-Computern und dem C 128 im 64er-Modus. Als Laufwerke sind die 1541 und die 1571 von Commodore verwendbar.

Der Bildschirm bleibt während des Ladevorgangs ausgeschaltet, da sonst der VIC-Chip die schnellen Bus-Routinen aus dem Takt bringen würde. Das Blinken der roten Laufwerks-LED ist völlig normal, es zeigt jeweils einen übertragenen Programmblock an. Damit läßt sich optisch die Arbeit von Hypra-Load verfolgen.

Weitere, am seriellen Bus angeschlossene Geräte dürften die Übertragung nicht stören. Solange Hypra-Load aktiviert ist, führen Kassetten-Zugriffe zum System-Absturz!

Soll Hypra-Load abgeschaltet werden, genügt die Befehlsfolge »POKE 1,55«.

Der VERIFY-Befehl funktioniert nicht bei aktiviertem Hypra-Load.

Für diejenigen, die sich für die Programmierung des Autostarts von Hypra-Load interessieren: Er wurde durch Verbiegen des Stop-Vektors realisiert.

(Karsten Schramm/Boris Schneider/og)

Hypra-Save

Hypra-Save ist eine Ergänzung zu Hypra-Load. Es speichert Programme 3- bis 5mal schneller und kann mit Hypra-Load verwendet werden.

Ein großer Nachteil der Diskettenstation VC1541 ist die durch den seriellen Bus und durch das DOSV2.6 bedingte geringere Geschwindigkeit. Inzwischen gibt es mehrere Programme, die das Laden von Diskette beschleunigen. Mit der hier vorgestellten Routine geht jetzt auch das Speichern von Programmen mit dem C64 wesentlich rascher.

Hypra-Save ist 3- bis 5mal so schnell wie die Originalroutine. Es verträgt sich mit Hypra-Load und vielen anderen, auch professionellen, Programmen und Basic-Erweiterungen. Zur Bedienung von Hypra-Save sollten Sie folgendes beachten: Die Eingabe muß mit dem MSE erfolgen. Nach dem Laden startet man es wie gewohnt mit RUN. Danach sollte man NEW eingeben, wenn man ein eigenes Programm schreiben will.

Hypra-Save kann mit oder ohne Verify speichern.

Gibt man vor dem Filenamen als erstes Zeichen einen Stern ein, so wird nicht verifiziert. Feststellbar an bis zu 5mal schnelleren Speicherzeiten. Mit Verify ist Hypra-Save etwa 3mal schneller als die Original-SAVE-Routine. Wer einen »25, WRITE ERROR« bisher nur aus der Literatur kennt, der kann getrost ohne Verify arbeiten. Selbstverständlich kann man weiterhin Programme überschreiben. Dann ist der Klammeraffe mit anzugeben. So überschreibt der Befehl SAVE "*@:name",8 ein File, ohne die auf Diskette geschriebenen Blöcke zu prüfen, also ohne Verify. Hat der Computer alle Daten gesendet, wird im Gegensatz zur Original-SAVE-Routine nicht gewartet, bis das Laufwerk die Datei geschlossen hat. Dies macht sich besonders beim Überschreiben von Programmen bemerkbar. Die Floppystation arbeitet noch, während der Computer sich längst zurückgemeldet hat. Man darf die Diskette selbstverständlich nicht vor dem Erlöschen der roten LED aus dem Laufwerk nehmen.

Beim Speichern von Programmen mit dem Klammeraffen kommt die 1541 häufig ins »Schleudern«, wie Sie vielleicht

Hinweis zum Abtippen
Dieses Listing muß mit dem MSE eingegeben werden. Den MSE finden Sie in dieser Ausgabe auf Seite 136.

```
PROGRAMM : HYPRA-SAVE      0801 0D3F
------------------------------------

0801 : 0C 08 C1 07 9E 20 32 30   77
0809 : 36 32 00 00 00 78 A5 01   B5
0811 : 48 A9 34 85 01 A0 28 B9   15
0819 : 51 08 99 4F 01 88 D0 F7   46
0821 : A2 06 84 AC A9 D0 85 AD   0F
0829 : A9 A0 85 14 A9 08 85 15   21
0831 : B1 14 91 AC C8 D0 F9 E6   AF
0839 : 15 E6 AD CA D0 F2 20 6D   86
0841 : 01 68 85 01 58 A2 DA BD   79
0849 : A0 07 20 D2 FF E8 D0 F7   4A
0851 : 60 A5 BA C9 04 B0 03 4C   D6
0859 : ED F5 78 A6 01 A9 34 85   6D
0861 : 01 8E 9F D4 20 4F D4 AE   59
0869 : 9F D4 86 01 58 60 A9 50   04
0871 : 8D 32 03 A9 01 8D 33 03   5D
0879 : 60 0D 0D 09 0E 55 4C 54   2A
0881 : 52 41 53 41 56 45 20 36   ED
0889 : 34 20 41 4B 54 49 56 0D   8A
0891 : 49 4E 49 54 20 3A 20 53   D9
0899 : 59 53 20 33 36 35 0D BA   C1
08A1 : 8E A0 D4 AD 19 03 8D A1   8D
08A9 : D4 AD 18 03 8D A2 D4 A9   4F
08B1 : 9C 8D 18 03 A9 C1 8D 19   8B
08B9 : 03 AD 9F D4 85 01 AD 00   2C
08C1 : DD 8D 1F 01 58 A9 61 85   B0
08C9 : B9 A2 2C A0 00 B1 BB C9   03
08D1 : 2A D0 0A A2 4C C6 B7 E6   E2
08D9 : BB D0 02 E6 BC 8E A5 C3   B8
08E1 : A9 C1 48 A9 75 48 A5 B7   52
08E9 : D0 03 4C 10 F7 20 D5 F3   10
08F1 : 20 8F F6 A5 BA 20 0C ED   04
08F9 : A5 B9 20 B9 ED A9 00 85   F1
0901 : 90 20 DD ED 20 DD ED 20   BF
0909 : FE ED A5 90 F0 02 18 60   B9
0911 : A9 1A A2 C2 85 AC 86 AD   FB
0919 : A9 46 A2 01 85 14 86 15   EB
0921 : A2 05 A5 BA 20 0C ED A9   74
0929 : 6F 20 B9 ED A0 FD B9 15   DF
0931 : C1 20 DD ED C8 D0 F7 A5   76
0939 : 14 20 DD ED A5 15 20 DD   D2
0941 : ED A9 1E 20 DD ED A0 00   5E
0949 : B1 AC 20 DD ED C8 C0 1E   78
0951 : 90 F6 20 FE ED A5 AC 18   33
0959 : 69 1E 85 AC 90 02 E6 AD   D8
0961 : A5 14 18 69 1E 85 14 90   C3
0969 : 02 E6 15 CA D0 B4 A5 BA   3C
0971 : 20 0C ED A9 6F 20 B9 ED   03
0979 : A0 FB B9 1A C1 20 DD ED   39
0981 : C8 D0 F7 20 FE ED 78 A2   3A
0989 : 00 8E 00 DD AD 11 D0 29   85
0991 : EF 8D 11 D0 BD AE C2 9D   3D
0999 : 00 C5 E8 D0 F7 AD 00 DD   79
09A1 : 30 FB 20 FE C1 A2 00 BD   63
09A9 : AE C3 9D 00 C5 E8 D0 F7   77
09B1 : 20 FE C1 20 8E FB A2 01   1A
09B9 : A5 AC 9D 01 C5 E8 A5 AD   D1
09C1 : 9D 01 C5 E8 20 D1 FC B0   53
09C9 : 38 A5 AD C9 C0 90 1D C9   11
09D1 : C6 B0 19 69 16 85 AD A9   FA
09D9 : 34 85 01 B1 AC 48 AD 9F   49
09E1 : D4 85 01 A5 AD E9 15 85   F6
09E9 : AD 68 B0 02 B1 AC 9D 01   30
09F1 : C5 E8 20 DB FC E6 [illegible] 90   A6
09F9 : CB 20 D1 FC B0 03 [illegible] FF   B2
0A01 : 2C A9 00 48 8D 00 C5 8E   18
0A09 : 01 C5 20 FE C1 A2 01 68   DB
0A11 : D0 B2 18 A9 00 48 AD 11   91
0A19 : D0 09 10 8D 11 D0 AD 1F   B0
0A21 : 01 8D 00 DD 78 A9 34 85   55
0A29 : 01 AD A2 D4 8D 18 03 AD   45
0A31 : A1 D4 8D 19 03 68 AE A0   32
0A39 : D4 9A 60 38 AD 1F 01 09   63
0A41 : 10 8D 1F 01 B0 CD B9 00   60
0A49 : C5 85 95 A2 00 2C 00 DD   A8
0A51 : 50 FB 2C 00 DD 10 05 E8   EE
0A59 : D0 F8 F0 B6 A9 10 8D 00   0A
0A61 : DD AA 66 95 6A 66 95 6A   E5
0A69 : 4A 4A 8D 00 DD 8A 66 95   33
0A71 : 6A 66 95 6A 4A 4A 8D 00   EE
0A79 : DD 8A 66 95 6A 66 95 6A   ED
0A81 : 4A 4A 8D 00 DD 8A 66 95   4B
0A89 : 6A 66 95 6A 4A 4A 8D 00   06
0A91 : DD EA EA EA A9 00 8D 00   CC
0A99 : DD C8 D0 AA 60 A0 00 98   A0
0AA1 : 59 00 C5 C8 D0 FA 85 14   A8
0AA9 : 20 A7 C1 88 A5 14 4C AA   A0
0AB1 : C1 4D 2D 57 4D 2D 45 B7   12
0AB9 : 01 A0 00 84 11 A9 02 8D   1C
0AC1 : 00 18 A9 04 2C 00 18 F0   BD
0AC9 : FB A9 00 8D 00 18 A2 03   9C
0AD1 : CA D0 FD A2 0A AD 00 18   15
0AD9 : 4A 6A 4A 66 85 0A 0A 66   55
0AE1 : 85 AD 00 18 4A 6A 4A 66   2E
0AE9 : 85 0A 0A 66 85 AD 00 18   B9
0AF1 : 4A 6A 4A 66 85 0A 0A 66   6D
0AF9 : 85 AD 00 18 8E 00 18 4A   36
0B01 : 6A 4A 66 85 0A 0A 66 85   70
0B09 : A5 85 91 30 45 11 85 11   F0
0B11 : C8 D0 AA 88 60 85 31 20   34
0B19 : 46 01 B1 30 48 20 4A 01   03
0B21 : 68 91 30 A5 11 F0 ED 4C   FB
0B29 : 43 E8 78 A9 0A 8D 00 18   71
0B31 : A2 00 88 D0 FD CA D0 FA   7F
0B39 : A9 0A 85 69 A9 00 85 30   87
0B41 : A9 03 20 A2 01 A9 04 20   76
0B49 : A2 01 4C 3D 04 A5 00 A2   D9
0B51 : 01 86 00 29 02 F0 10 A6   F0
0B59 : 98 86 32 20 2E 04 20 AF   A8
0B61 : 03 20 03 04 4C 69 F9 A2   F3
0B69 : 08 20 35 03 A2 0A 20 35   94
0B71 : 03 A2 08 20 F5 03 A2 0A   E2
0B79 : 20 F5 03 A5 8C 30 E8 4C   90
0B81 : 69 F9 86 32 86 98 A5 8C   AC
0B89 : 10 61 B5 83 D0 5D BD 31   79
0B91 : 04 20 A2 01 C8 A6 32 A9   4C
0B99 : 80 95 83 A5 80 95 00 A5   79
0BA1 : 81 95 01 B1 30 F0 1C 20   9F
0BA9 : 21 F1 A6 82 F6 B5 D0 02   21
0BB1 : F6 BB A0 00 A5 80 91 30   B2
0BB9 : C8 A5 81 91 30 A5 80 C5   A4
0BC1 : 22 F0 02 84 8C A6 32 A9   87
0BC9 : 00 85 30 85 33 85 2E 85   6C
0BD1 : 36 85 0C 85 50 A9 BB 85   CA
0BD9 : 34 BD 31 04 85 2F 20 E9   DF
0BE1 : F5 85 3A BD 32 04 20 A3   EA
0BE9 : F7 A6 32 B5 83 F0 52 20   C0
0BF1 : 2E 04 AD 00 1C 29 10 D0   7A
0BF9 : 03 4C 81 F5 20 10 F5 A2   E1
0C01 : 09 50 FE B8 CA D0 FA A9   7B
0C09 : FF 8D 03 1C AD 0C 1C 29   11
0C11 : 1F 09 C0 8D 0C 1C A9 FF   DF
0C19 : A2 05 8D 01 1C B8 50 FE   88
0C21 : B8 CA D0 FA A0 BB B1 0C   98
0C29 : 50 FE B8 8D 01 1C C8 D0   8E
0C31 : F5 B1 30 50 FE B8 8D 01   03
0C39 : 1C C8 D0 F5 50 FE 4C 00   DA
0C41 : FE 60 86 32 2C 24 04 86   58
0C49 : 98 B5 83 F0 F4 20 2E 04   CC
0C51 : 20 0A F5 A0 BB B1 0C 50   22
0C59 : FE B8 4D 01 1C D0 19 C8   65
0C61 : D0 F3 B1 30 50 FE B8 4D   18
0C69 : 01 1C D0 0C C8 C0 FD D0   5A
0C71 : F1 A6 32 A9 00 95 83 60   F3
0C79 : 4C C5 F6 BD 31 04 85 31   C9
0C81 : BD 32 04 85 0D 60 05 01   F3
0C89 : 06 04 58 20 19 F1 A9 84   7C
0C91 : D5 A7 F0 05 95 A7 20 42   B2
0C99 : D0 A9 40 8D F9 02 A9 01   58
0CA1 : 85 83 20 07 D1 90 03 4C   17
0CA9 : F8 CF 20 3E DE F6 B5 A9   28
0CB1 : 00 85 8B 85 8D A9 80 85   3B
0CB9 : 8C A5 80 85 06 A9 E0 85   25
0CC1 : 00 A5 00 30 FC F0 24 C9   15
0CC9 : 01 F0 EA A5 18 85 06 A5   C3
0CD1 : 19 85 07 A2 00 A9 B0 20   13
0CD9 : 7D D5 20 99 D5 A9 E2 20   F3
0CE1 : 7D D5 20 99 D5 A5 8C D0   E2
0CE9 : CC F0 CE 4C 23 DB A0 C0   80
0CF1 : A2 D6 20 6A D4 A0 D0 A2   2F
0CF9 : C0 20 6A D4 20 00 C0 48   94
0D01 : A0 D6 A2 C0 20 6A D4 68   46
0D09 : 60 AD FA FF 48 AD FB FF   E0
0D11 : 48 A9 FB 8D FA FF A9 D3   DC
0D19 : 8D FB FF 84 AD 86 15 A0   D9
0D21 : 00 84 AC 84 14 A2 06 B1   F1
0D29 : AC 91 14 C8 D0 F9 E6 AD   90
0D31 : E6 15 CA D0 F2 68 8D FB   0F
0D39 : FF 68 8D FA FF 60 B2 4C   95
```

Listing zu »Hypra-Save«. Beachten Sie bitte die Eingabehinweise auf Seite 136.

aus eigener Erfahrung wissen. So kann es passieren, daß einige Programme nicht mehr geladen werden können. Löschen Sie daher ein Programm erst mit dem SCRATCH-Befehl und speichern Sie erst dann die neue Version.

Anstelle von RUN/STOP dient bei Hypra-Save die RESTORE-Taste zum Abbrechen. Allerdings wird die Programmdatei dann nicht geschlossen und erscheint im Directory mit einem Stern. Möchte man die nicht geschlossene Datei löschen, so geht das nicht mit dem SCRATCH-, sondern nur mit dem VALIDATE-Befehl. Übrigens blinkt die rote LED nach dem Drücken von RESTORE. Liest man dann den Fehlerkanal aus, so erhält man die Meldung »51, OVERFLOW IN RECORD«, gefolgt von Track und Sektor des letzten Blocks. Diese Fehlermeldung weist sonst auf einen Übertragungsfehler hin, der von zu vielen Geräten am seriellen Bus herrühren kann.

Sollte man eine merkwürdige Fehlernummer wie zum Beispiel 61 oder 71 erhalten, hilft meist nur ein Aus- und Einschalten der Floppy. Nach RUN/STOP-RESTORE oder nach einem Reset ist der SAVE-Vektor zurückgesetzt. Hypra-Save läßt sich dann mit SYS 365 wieder aktivieren.

(Martin Pfost/og)

Der Sprite-Editor

Wollen Sie sich etwas näher mit den Sprites beschäftigen und experimentieren? Dann ist unsere Sprite-Bibliothek genau das richtige für Sie!

Wir haben für Sie ein Programm vorbereitet, welches es Ihnen ermöglicht, eigene Sprites zu entwickeln und zu verändern. Zusätzlich besteht die Möglichkeit, Sprites aus fremden Programmen zu betrachten und zu manipulieren. Wenn Sie Listing 1 mit Hilfe des MSE (zu finden in diesem Heft) abgetippt haben, können Sie es unter dem Namen »EDE.MA« auf Diskette speichern. Der eigentliche Sprite-Editor ist in Listing 2 abgedruckt (bitte unter dem Namen »EDE« speichern). Bevor Sie mit dem Sprite-Editor arbeiten, sollten Sie den Computer kurz aus- und einschalten und das Ladeprogramm (Listing 3) neu in den Computer laden. Dies ist notwendig, da der Sprite-Editor nicht an der üblichen Stelle im Speicher abgelegt wird, sondern ab Adresse 16384 arbeitet.

Nachdem Sie das Ladeprogramm gestartet haben, werden der Editor und die Assemblerroutinen geladen. In der linken Hälfte des Bildschirms erscheint nun das Bitmuster des aktuellen Sprites. In der rechten oberen Ecke wird eine Zahlenreihe ausgedruckt, die den Zustand der acht Sprites darstellt. Das aktuelle Sprite wird durch eine weiße Ziffer gekennzeichnet, eine revers dargestellte Ziffer repräsentiert ein eingeschaltenes Sprite. Unter der Zahlenreihe wird der Block, aus dem die Form des aktuellen Sprites gelesen wird, angezeigt. »Sprite«, »Multi1« und »Multi2« zeigen die Spritefarbe und die beiden Multicolor-Farben. Im »Work-Modus«, in dem Sie sich jetzt befinden, können Sie die folgenden Funktionen abrufen:

0 bis 7: wählen das aktuelle Sprite an.

+ und -: schalten aktuelles Sprite ein beziehungsweise aus.

Die Cursortasten : bewegen das Sprite über den Bildschirm.

F1, F3, F5: verändern die Spritefarben.

X: vergrößert das Sprite in X-Richtung (SHIFT X macht Vergrößerung wieder rückgängig).

Y: analog zu X, jedoch in Y-Richtung.

M: schaltet den Multicolor-Modus ein (SHIFT M ändert den Zustand wieder in den Singlecolor-Modus).

B: zählt die Blocknummer aufwärts (Shift B zählt abwärts). Bitte beachten Sie, daß Sie die Blöcke 0 bis 31 sowie die Blöcke 64 bis 127 nicht benutzen können.

A: Mit dieser Funktion können Sie die Werte zur Sprite-Animation festlegen. Die erscheinende Zahlenreihe zeigt die Reihenfolge der Sprites, in der sie beim Aufruf der Animations-Funktion erscheinen. Hierauf müssen Sie die Anzahl der Bewegungsphasen eingeben (maximal 9) und die entsprechenden Sprites, welche erscheinen sollen. Zu guter Letzt können Sie die Verzögerung bestimmen, mit welcher die Animation ablaufen soll (0 bis 9). Die Eingabe kann vorzeitig unterbrochen werden, indem Sie die RETURN-Taste drücken (Die RETURN-Taste ohne Eingabe bewirkt übrigens auch bei allen weiteren Funktionen einen vorzeitigen Abbruch).

SHIFT A: ruft die Animations-Routine auf (Um Sie wieder zu stoppen, müssen Sie irgendeine Taste drücken).

C: Hiermit ist es möglich, einen Block in einen anderen zu kopieren. Der Quellblock wird in den Zielblock kopiert, sobald Sie die entsprechenden Daten eingegeben haben.

D: gibt das Directory Ihrer Diskette auf dem Bildschirm aus.

S: ermöglicht es Ihnen, Ihre Spritedaten auf Diskette zu speichern.

L: lädt die Spritedaten wieder in den Speicher zurück.

P: erzeugt aus Ihren Spritedaten entsprechende DATA-Zeilen. Zu Beginn der Funktion werden Sie nach dem Block gefragt, welcher in DATA-Zeilen umgewandelt werden soll. Wenn Sie nun die Zeilennummer, mit der die DATA-Zeilen beginnen, und den Abstand der Zeilen eingeben, wird ein eigenständiges Basic-Programm generiert.

SHIFT P : stoppt den Sprite-Editor und legt den Basic-Programmstart auf die Adresse, in der Ihre DATA-Zeilen erzeugt wurden. Sie könnten dieses Programm nun speichern oder erweitern. Es gibt jedoch auch die Möglichkeit, in den Sprite-Editor zurückzukehren und weitere Sprites in Ihr Programm aufzunehmen. Geben Sie »POKE 44,64:RUN« ein und alles ist wieder beim alten.

F1, F3, F5: verändern die Spritefarben.

F7: Mit dieser Taste kommen Sie aus dem »Work-Modus« in den »Design-Modus« und umgekehrt. Im »Design-Modus« ist es möglich, die Form des Sprites zu verändern, indem Sie mit den Cursortasten den in der linken oberen Ecke erscheinenden Stern auf dem Bitmuster Ihres Sprites umherbewegen (HOME, CLR und RETURN behalten ihre ursprüngliche Funktion). **Aber Achtung!** Versuchen Sie nicht, die Sprite-Blöcke 0 bis 32 zu ändern, da Sie hierbei direkt in Speicherbereiche eingreifen, die vom Betriebssystem benötigt werden. Änderungen können den Computer zum Absturz bringen! Hier nun die weiteren Befehle:

F1, F3, F5: setzen einen Punkt in der entsprechenden Farbe (SPRITE, MULTI1, MULTI2). Falls Ihr Sprite im Singlecolor-Modus dargestellt wird, funktioniert natürlich nur F1.

F2: löscht einen Punkt.

[und]: verschieben das gesamte Bitmuster nach links beziehungsweise nach rechts.

@ und /: analog zu oben, jedoch in vertikaler Richtung.

X: spiegelt das Bitmuster an der X-Achse

Y: spiegelt an der Y-Achse

(Thomas Erhart/dm)

Hinweis zum Abtippen
Dieses Listing muß mit dem MSE eingegeben werden.
Den MSE finden Sie in dieser Ausgabe auf Seite 136.

```
PROGRAMM : EDE.MA              C030 C289
----------------------------------------

C030 : A9 04 85 FE A9 00 8D 35   58
C038 : 03 8D 36 03 A9 29 85 FD   E6
C040 : A9 80 8D 34 03 AC 35 03   B4
C048 : B1 FB 2D 34 03 AA 8C 35   EB
C050 : 03 AC 36 03 A9 A0 E0 00   BA
C058 : D0 02 A9 2E 91 FD C8 8C   9F
C060 : 36 03 C0 18 D0 12 A9 00   8F
C068 : 8D 36 03 18 A5 FD 69 28   14
C070 : 85 FD A5 FE 69 00 85 FE   E8
C078 : AD 34 03 18 6A 8D 34 03   ED
C080 : 90 C3 EE 35 03 AC 35 03   C5
C088 : C0 3F D0 B4 60 EA EA EA   91
C090 : EA EA EA EA A9 29 85 FD   FD
C098 : A9 04 85 FE A0 00 A2 00   19
C0A0 : B1 FD C9 A0 D0 02 E8 E8   69
C0A8 : C8 B1 FD C9 A0 D0 01 E8   68
C0B0 : E0 00 D0 02 A9 2E E0 01   96
C0B8 : D0 02 A9 B2 E0 02 D0 02   AF
C0C0 : A9 B1 E0 03 D0 02 A9 B3   05
C0C8 : 91 FD 88 91 FD C8 C8 C0   77
C0D0 : 18 D0 CB 18 A5 FD 69 28   86
C0D8 : 85 FD A5 FE 69 00 85 FE   50
C0E0 : C9 07 D0 B8 A5 FD C9 71   CC
C0E8 : D0 B2 60 EA EA EA A0 00   0F
C0F0 : C8 C8 B1 FB 29 01 C9 00   CA
C0F8 : 18 F0 01 38 88 88 B1 FB   5B
C100 : 6A 91 FB C8 B1 FB 6A 91   12
C108 : FB C8 B1 FB 6A 91 FB C8   08
C110 : C0 3F D0 DC 60 EA A0 3E   9C
C118 : 88 88 B1 FB 29 80 C9 00   8E
C120 : 18 F0 01 38 C8 C8 B1 FB   89
C128 : 2A 91 FB 88 B1 FB 2A 91   F1
C130 : FB 88 B1 FB 2A 91 FB 88   8B
C138 : C0 FF D0 DC 60 EA A0 3E   24
C140 : B1 FB 99 C0 02 88 C0 3B   4B
C148 : D0 F6 B1 FB C8 C8 C8 91   98
C150 : FB 88 88 88 88 C0 FF D0   F3
C158 : F1 C8 B9 FC 02 91 FB C0   D9
C160 : 02 D0 F6 60 EA EA A0 00   1C
C168 : B1 FB 99 C0 02 C8 C0 03   05
C170 : D0 F6 B1 FB 88 88 88 91   B9
C178 : FB C8 C8 C8 C8 C0 3F D0   54
C180 : F1 88 B9 84 02 91 FB C0   D2
C188 : 3C D0 F6 60 EA EA A0 00   7E
C190 : 8C 39 03 A2 3C B1 FB 9D   4A
C198 : C0 02 C8 E8 EE 39 03 AD   C9
C1A0 : 39 03 C9 03 D0 0B A9 00   3A
C1A8 : 8D 39 03 CA CA CA CA CA   B0
C1B0 : CA C0 3F D0 E0 A0 00 B9   4B
C1B8 : C0 02 91 FB C8 C0 3F D0   8E
C1C0 : F6 60 00 00 A0 00 B1 FB   AF
C1C8 : C8 C8 AA B1 FB 8D C0 02   08
C1D0 : 8A 91 FB 88 88 AD C0 02   30
C1D8 : 91 FB C8 C8 C8 C0 3F D0   E3
C1E0 : E5 A0 00 A2 08 B1 FB 4A   FC
C1E8 : 91 FB 8C 39 03 8A A8 AE   46
C1F0 : 39 03 3E C0 02 8C 39 03   C2
C1F8 : 8A A8 AE 39 03 CA E0 00   B3
C200 : D0 E3 C8 C0 3F D0 DC 18   2A
C208 : 90 AB EA EA EA EA EA EA   0D
C210 : A9 24 85 FB A9 FB 85 BB   B4
C218 : A9 00 85 BC A9 01 85 B7   E2
C220 : A9 08 85 BA A9 60 85 B9   AD
C228 : 20 D5 F3 A5 BA 20 B4 FF   64
C230 : A5 B9 20 96 FF A9 00 85   E5
C238 : 90 A0 03 84 FB 20 A5 FF   C1
C240 : 85 FC A4 90 D0 2F 20 A5   D1
C248 : FF A4 90 D0 28 A4 FB 88   80
C250 : D0 E9 A6 FC 20 CD BD A9   19
C258 : 20 20 D2 FF 20 A5 FF A6   B9
C260 : 90 D0 12 AA F0 06 20 D2   98
C268 : FF 18 90 F0 A9 0D 20 D2   DF
C270 : FF A0 02 D0 C6 20 42 F6   BE
C278 : 60 EA EA EA EA EA A9 20   52
C280 : 20 20 20 20 20 20 20 20   80
C288 : 20 FF FF FF FF FF FF FF   A8
```

Listing 1: »EDE.MA«. Bitte mit dem MSE eingeben.

Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel {CLR}, muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksummer 64 auf Seite 135.

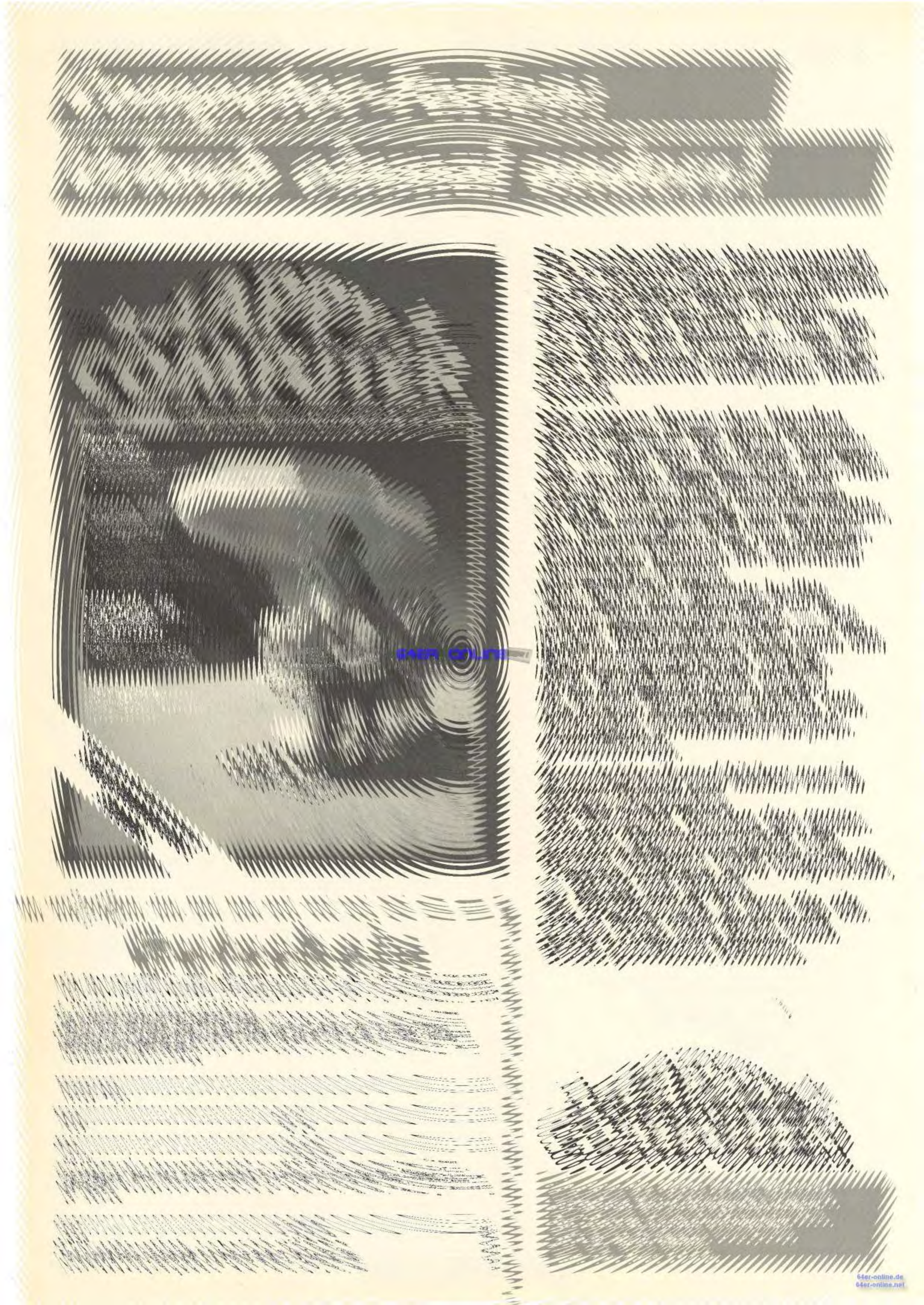

```
9 REM ------------------                  <168>
10 REM 49200 NORM                         <133>
11 REM 49300 MULTI                        <101>
12 REM 49390 RIGHT                        <210>
13 REM 49430 LEFT                         <122>
14 REM 49470 DOWN                         <156>
15 REM 49510 UP                           <128>
16 REM 49550 MIRRO Y                      <102>
17 REM 49604 MIRRO X                      <040>
18 REM 49680 DIRECTO                      <074>
19 REM ------------------                 <178>
50 IF PEEK(2)>64 THEN POKE 46,PEEK(2):CLR <151>
51 POKE 56,100                            <182>
100 POKE 53280,11                         <092>
101 POKE 53281,11                         <221>
102 V=53248:POKE 650,128                  <012>
103 FOR AA=0 TO 15 STEP 2 :POKE V+AA,0:POK
    E V+AA+1,100:NEXT:POKE V+16,255       <022>
104 FOR AA=2040 TO 2047:POKE AA,S(AA-2040)
    :NEXT                                 <192>
110 PRINT"{CLR,LIG.GREEN}OYYYYYYYYYYYYYYY
    YYYYYYYYP"                            <188>
115 FOR AA=0 TO 20:PRINT"G................
    ........H":NEXT                       <148>
120 PRINT"LPPPPPPPPPPPPPPPPPPPPPPPPP@"      <136>
130 PRINT"{HOME,2DOWN}"TAB(29)"{RVSON,SPAC
    E}SPRITE ":IF SP>7 THEN SP=7          <218>
131 PRINT TAB(29)"{RVSON,SPACE}MULTI1 "   <217>
132 PRINT TAB(29)"{RVSON,SPACE}MULTI2 "   <026>
133 PRINT"{HOME,DOWN}"TAB(27)"BLOCK :{4SPA
    CE,4LEFT}"S(SP)                       <084>
134 PRINT"{HOME,5DOWN}"TAB(29)"{8SPACE}";:
    MM=1                                  <249>
135 IF (PEEK(V+28)AND(2↑SP))<>0 THEN PRINT
    "{8LEFT,SPACE}MULTIC ":MM=2           <039>
136 PRINT"{HOME}"TAB(29)"01234567"        <214>
140 FOR AA=29 TO 36                       <016>
141 POKE 55376+AA,PEEK(V+39+SP)           <245>
142 POKE 55416+AA,PEEK(V+37)              <199>
143 POKE 55456+AA,PEEK(V+38)              <203>
144 IF PEEK(198)<>0 THEN 146              <133>
145 IF (PEEK(V+21)AND(2↑(AA-29)))<>0 THEN
    POKE 1024+AA,PEEK(1024+AA)OR 128      <038>
146 NEXT                                  <156>
147 POKE 55296+29+SP,1                    <197>
150 AD=S(SP)*64:AH=INT(AD/256):AL=AD-AH*25
    6                                     <253>
151 POKE 251,AL:POKE 252,AH               <229>
152 SYS 49200:IF (PEEK(V+28)AND(2↑SP))<>0
    THEN SYS 49300                        <177>
180 IF M=1 THEN 300                       <175>
190 GOTO 290                              <014>
200 GET K$:IF K$=""THEN 200               <190>
205 IF K$="{F7}"THEN M=1:GOTO 299         <116>
210 IF K$="+"THEN POKE V+21,PEEK(V+21)OR(2
    ↑SP):GOTO 136                         <088>
211 IF K$="-"THEN POKE V+21,PEEK(V+21)AND(
    255-(2↑SP)):GOTO 136                  <136>
212 IF K$="X"THEN POKE V+29,PEEK(V+29)OR(2
    ↑SP):GOTO 200                         <215>
213 IF K$="X"THEN POKE V+29,PEEK(V+29)AND(
    255-(2↑SP)):GOTO 200                  <028>
214 IF K$="Y"THEN POKE V+23,PEEK(V+23)OR(2
    ↑SP):GOTO 200                         <022>
215 IF K$="Y"THEN POKE V+23,PEEK(V+23)AND(
    255-(2↑SP)):GOTO 200                  <091>
216 IF K$="M"THEN POKE V+28,PEEK(V+28)OR(2
    ↑SP):GOTO 134                         <010>
217 IF K$="M"THEN POKE V+28,PEEK(V+28)AND(
    255-(2↑SP)):GOTO 134                  <127>
220 IF VAL(K$)<>0 OR K$="0" THEN SP=VAL(K$
    ):GOTO 130                            <070>
221 IF K$="{UP}"THEN POKE V+SP*2+1,ABS(PEE
    K(V+SP*2+1)-1):IF PEEK(197)<>64 THEN 2
    21                                    <153>
222 IF K$="{DOWN}"THEN POKE V+SP*2+1,ABS(P
    EEK(V+SP*2+1)+1):IF PEEK(197)<>64 THEN
     222                                  <096>
223 IF K$="{LEFT}"THEN POKE V+SP*2,ABS(PEE
    K(V+SP*2)-1):IF PEEK(197)<>64 THEN 223 <221>
224 IF K$="{RIGHT}"THEN POKE V+SP*2,ABS(PE
    EK(V+SP*2)+1):IF PEEK(197)<>64 THEN 22
    4                                     <255>
225 IF K$="B"THEN S(SP)=S(SP)+1:GOTO 228  <195>
226 IF K$="B"THEN S(SP)=S(SP)-1:GOTO 228  <036>
227 GOTO 230                              <211>
228 IF S(SP)<11 OR S(SP)>255 THEN S(SP)=S(
    SP)-SGN(S(SP)-50)                     <146>
229 PRINT"{HOME,DOWN}"TAB(34)S(SP):POKE 20
    40+SP,S(SP):IF PEEK(197)<>64 THEN 225 <084>
230 IF K$="{F1}"THEN POKE V+SP+39,(PEEK(V+
    SP+39)+1)AND 15:GOTO 140              <142>
231 IF K$="{F3}"THEN POKE V+37,(PEEK(V+37)
    +1)AND 15:GOTO 140                    <071>
```

Listing 2: Der Sprite-Editor »EDE«.

```
232 IF K$="{F5}"THEN POKE V+38,(PEEK(V+38)
    +1)AND 15:GOTO 140                       <074>
240 IF K$="A"THEN 500                        <001>
241 IF K$="A"THEN 550                        <106>
242 IF K$="D"THEN 600                        <137>
243 IF K$="L"THEN 610                        <150>
244 IF K$="S"THEN 610                        <026>
245 IF K$="P"THEN PRINT"{CLR}":GOTO 650      <220>
246 IF K$="C"THEN 800                        <020>
247 IF K$="P"THEN POKE 2,PEEK(46):POKE 56,
    160:POKE 44,100:POKE 46,150:CLR:END      <039>
280 IF K$="B"OR K$="B"THEN 150               <066>
290 PRINT"{HOME,23DOWN}"TAB(30)" WORK "      <137>
298 GOTO 200                                 <236>
299 PRINT"{HOME,23DOWN}"TAB(30)"DESIGN":X=
    0:Y=0:I=PEEK(1065)                       <242>
300 GET K$:IF K$=""THEN 300                  <068>
305 IF K$="{F7}"THEN M=0:POKE 1065+X+Y*40,
    I:GOTO 290                               <038>
310 IF K$="{UP}" THEN YR=-1                  <252>
311 IF K$="{DOWN}" THEN YR=+1                <157>
312 IF K$="{LEFT}" THEN XR=-1                <000>
313 IF K$="{RIGHT}" THEN XR=+1               <161>
315 IF YR<>0 OR XR<>0 THEN 360               <211>
320 IF K$=";"THEN FOR AA=1 TO MM:SYS 49390
    :NEXT:GOTO 150                           <071>
321 IF K$=":"THEN FOR AA=1 TO MM:SYS 49430
    :NEXT:GOTO 150                           <006>
322 IF K$="@"THEN SYS 49510:GOTO 150         <242>
323 IF K$="/"THEN SYS 49470:GOTO 150         <203>
324 IF K$="X"THEN SYS 49604:GOTO 150         <226>
325 IF K$="Y"THEN SYS 49550:GOTO 150         <130>
335 IF K$="{F1}"THEN I=160+(MM-1)*17:F=1:G
    OTO 400                                  <046>
336 IF K$="{F2}"THEN I=46 :F=0:GOTO 400      <062>
337 IF MM=1 THEN 340                         <143>
338 IF K$="{F3}"THEN I=178:F=2:GOTO 400      <020>
339 IF K$="{F5}"THEN I=179:F=3:GOTO 400      <217>
340 IF K$="{CLR}"THEN FOR AA=0 TO 63:POKE
    S(SP)*64+AA,0:NEXT:GOTO 150              <091>
341 IF K$="{HOME}"THEN XR=-X/MM:YR=-Y        <034>
342 IF K$=CHR$(13) THEN XR=-X/MM:YR=1        <131>
355 IF YR=0 AND XR=0 THEN 300                <006>
360 POKE 1065+X+Y*40,I:IF MM=2 THEN POKE 1
    066+X+Y*40,I                             <046>
362 X=X+XR*MM:XR=0:IF X>23 THEN X=0:YR=1     <194>
363 IF X<0 THEN X=24-MM:YR=-1                <155>
364 Y=Y+YR:YR=0:IF Y>20 THEN Y=0             <124>
365 IF Y<0 THEN Y=20                         <019>
366 I=PEEK(1065+X+Y*40)                      <103>
367 POKE 1065+X+Y*40,42                      <218>
380 GOTO 300                                 <070>
400 AD=S(SP)*64+Y*3+X/8                      <035>
420 AP=7-(X/8-INT(X/8))*8                    <142>
425 XR=1                                     <239>
430 IF MM=1 THEN 480                         <245>
440 POKE AD,PEEK(AD)AND(255-(2↑AP+2↑(AP-1)
    ))                                       <205>
450 POKE AD,PEEK(AD)OR((F AND 1)*2↑AP+((F
    AND 2)/2)*2↑(AP-1))                      <046>
460 GOTO 355                                 <134>
480 POKE AD,PEEK(AD)AND(255-(2↑AP))          <164>
490 POKE AD,PEEK(AD)OR((F AND 1)*2↑AP)       <066>
495 GOTO 355                                 <169>
500 BE=PEEK(V+21):POKE V+21,0                <064>
505 PRINT"{HOME,7DOWN}"TAB(28);:FOR AA=1 T
    O 9:PRINT RIGHT$(STR$(BS(AA)),1);:NEXT   <036>
506 PRINT                                    <098>
510 PRINT TAB(28)"{DOWN}ANZAHL :";           <139>
511 GET K$:IF K$=""THEN 511                  <023>
512 IF K$=CHR$(13) THEN 540                  <134>
513 BA=VAL(K$):IF BA=0 THEN 511              <210>
514 PRINT BA:PRINT                           <063>
515 FOR W=1 TO BA                            <163>
520 PRINT TAB(28)"SP";W;"{LEFT}:";           <004>
521 GET K$:IF K$=""THEN 521                  <099>
522 IF K$=CHR$(13) THEN 540                  <144>
523 BS(W)=VAL(K$):IF BS(W)>7 THEN 521        <097>
524 PRINT BS(W)                              <043>
525 NEXT                                     <027>
530 PRINT TAB(28)"{DOWN}VERZOEGER";          <078>
531 GET K$:IF K$=""THEN 531                  <173>
532 IF K$=CHR$(13) THEN 540                  <154>
533 BV=VAL(K$)                               <220>
540 POKE V+21,BE                             <078>
541 PRINT"{HOME,6DOWN}":FOR W=0 TO 14        <030>
542 PRINT TAB(28)"{11SPACE}"                 <076>
543 NEXT                                     <045>
544 GOTO 200                                 <097>
550 BE=PEEK(V+21):POKE V+21,1                <124>
560 FOR AA=1 TO BA                           <016>
570 POKE 2040,S(BS(AA))                      <213>
575 FOR W=0 TO BV*100:NEXT                   <092>
580 NEXT                                     <082>
585 GET K$:IF K$=""THEN 560                  <036>
587 POKE V+21,BE                             <125>
588 POKE 2040,S(0)                           <150>
590 GOTO 200                                 <018>
600 PRINT"{CLR}"                             <080>
601 SYS 49680                                <247>
602 GET K$:IF K$=""THEN 602                  <213>
603 GOTO 110                                 <039>
610 PRINT"{CLR}"                             <090>
620 N$="":INPUT"NAME ";N$                    <033>
621 IF N$=""THEN 110                         <144>
625 IF K$="S"THEN 650                        <187>
630 OPEN 2,8,2,N$+",P,R"                     <138>
631 GET#2,N1$:IF N1$=""THEN 633              <142>
632 N1=ASC(N1$)                              <143>
633 GET#2,N2$:IF N2$=""THEN 635              <065>
634 N2=ASC(N2$)                              <154>
635 CLOSE 2                                  <146>
636 IF (N1=199) AND (N2=199) THEN 640        <110>
638 LOAD N$,8,1                              <094>
640 OPEN 1,8,15:INPUT#1,F1$,F2$,F3$,F4$      <213>
641 PRINT"{RVSON,2SPACE}"F2$"{2SPACE,DOWN}
    "                                        <200>
642 CLOSE 1:GOTO 620                         <059>
650 BS=0:INPUT"STARTBLOCK ";BS               <028>
651 IF BS=0 THEN 110                         <140>
652 IF BS<1 OR BS>255 THEN PRINT"{2UP}":GO
    TO 650                                   <211>
653 IF K$="P"THEN 700                        <050>
655 BE=0:INPUT"ENDBLOCK{3SPACE}";BE          <080>
656 IF BE=0 THEN 110                         <176>
657 IF BE<1 OR BE>255 THEN PRINT"{2UP}":GO
    TO 655                                   <224>
658 IF SGN(BE-BS)=-1 THEN PRINT"{3UP}":GOT
    O 650                                    <161>
660 OPEN 2,8,1,N$+",P,W"                     <044>
665 S=BS*64:SH=INT(S/256):SL=S-SH*256        <156>
666 PRINT#2,CHR$(SL);                        <136>
667 PRINT#2,CHR$(SH);                        <133>
670 FOR AA=S TO (BE+1)*64                    <010>
671 PRINT#2,CHR$(PEEK(AA));                  <172>
675 NEXT                                     <177>
678 CLOSE 2                                  <189>
680 GOTO 110                                 <116>
700 INPUT "ZEILENNUMMER";ZN                  <218>
710 INPUT "ABSTAND{5SPACE}";ZA               <152>
720 PRINT"{CLR,6DOWN}P056,160:P044,100:P04
    6,150:CLR:{2DOWN}"                       <123>
721 FOR AA=0 TO 3                            <224>
722 PRINT ZN+ZA*AA;"DATA";                   <030>
725 FOR AB=0 TO 15                           <092>
726 PRINT MID$(STR$(PEEK(BS*64+AB+AA*16))+
    ",",2,5);                                <134>
728 NEXT:PRINT"{LEFT,SPACE}":NEXT:PRINT"PO
    KE 44,64:RUN"                            <154>
730 FOR AA=0 TO 5:POKE 631+AA,13:NEXT        <250>
731 POKE 198,6                               <134>
732 PRINT"{HOME,3DOWN}"                      <152>
733 POKE 2,PEEK(46):END                      <130>
800 PRINT"{CLR}"                             <124>
810 INPUT"QUELLBLOCK ";QB                    <159>
811 IF QB=0 THEN 110                         <149>
815 INPUT"ZIELBLOCK{2SPACE}";QZ              <154>
816 IF QZ=0 THEN 110                         <027>
820 FOR AA=0 TO 63:POKE QZ*64+AA,PEEK(QB*6
    4+AA):NEXT                               <101>
825 GOTO 110                                 <007>
```

```
100 A=A+1:IF A=2 THEN 120                    <124>
110 LOAD "EDE.MA",8,1                        <029>
120 FOR AA=25600 TO 25604:POKE AA,0:NEXT     <085>
140 POKE 16384,0                             <240>
160 POKE 44,64:POKE 46,86:CLR                <007>
180 LOAD "EDE",8                             <237>
```

Listing 3: Das Ladeprogramm zu »EDE«.



Die Sprite-Bibliothek

Um Ihnen die Arbeit mit Sprites zu erleichtern, haben wir für Sie eine Sprite-Bibliothek entwickelt. Unser Programm soll Ihnen die Auswahl der Sprites für eigene Anwendungen erleichtern.

Wir haben für Sie ein Listing vorbereitet, mit dem Sie Sprites auf einfache Weise sichtbar machen können, ohne die Register des VICs per Hand zu manipulieren. Wenn Sie das Programm abgetippt haben, können Sie sich aus unserer Bibliothek die Sprites heraussuchen, die Sie gerne auf dem Bildschirm betrachten möchten. Beim Abtippen der Spritedaten ist es wichtig, daß die vorgegebene Zeilennumerierung eingehalten wird. Das Ende des Programms wird durch die Zahl 999 (in einer DATA-Zeile) angezeigt.

Starten Sie das Programm mit RUN, und auf dem Bildschirm erscheint das erste Sprite aus dem ersten DATA-Block (64 Zahlen). Am unteren Rand des Bildschirms steht Ihnen ein Menü zur Verfügung, mit dem Sie die einzelnen Funktionen des Programms aufrufen können. In der nachfolgenden Tabelle sind die Funktionstasten genau beschrieben:

F1: schaltet die Hintergrund- und die Rahmenfarbe um.
F3: ändert die Grundfarbe des Sprites.
F5 und F7: verändern die beiden Multicolorfarben des Sprites.
F2: schaltet zwischen Einfarb- und Mehrfarbmodus um.
F4: vergrößert und verkleinert das Sprite in X-Richtung.
F6: ist analog zu F4 für die Y-Richtung zuständig.
CR: Wenn Sie auf die RETURN-Taste drücken, wird das nächste Sprite angezeigt.
F8: Mit dieser Taste kommen Sie in den Korrekturmodus.

Im Korrekturmodus ist es möglich, die Sprite-Daten zu ändern und sofort wieder in DATA-Zeilen zu speichern. In diesem Modus werden die Werte des aktuellen Sprites auf dem Bildschirm angezeigt. Wenn Sie die Frage »Korrigieren« mit N (Nein) beantworten, kehrt das Programm wieder in das Grundmenü zurück. Möchten Sie das Sprite noch einmal überprüfen oder verändern, so können Sie das, ohne das Programm zu verlassen. In diesem Fall geben Sie ein J ein und der Cursor springt zu der ersten Rasterzeile. Sie können jetzt die betreffende Rasterzeile verändern (».« = gelöschtes Bit / »X« = gesetztes Bit) oder mit RETURN bestätigen. Damit springt der Cursor zur nächsten Position. Vorsicht, benutzen Sie bitte nie die Cursortasten, da das Programm mit einer einfachen INPUT-Routine arbeitet, die Ihr Sprite zerstören würde! Mit »£« und RETURN springt der Cursor wieder zum ersten Wert in der Tabelle, mit »↑« kommen Sie sofort in das Menü zurück. Sobald Sie mit der RETURN-Taste alle Daten bestätigt haben, erscheint wieder die alte Fußzeile. Für den Fall, daß die Daten verändert wurden, wird das alte Sprite mit »S« durch das neue Sprite ersetzt, indem die korrigierten DATA-Zeilen überschrieben werden. Darauf erscheint wieder das Menü-Bild.

Der Aufbau der Programms

Zeilen 110 bis 170: Hier werden die aktuellen Spritedaten aus dem Speicher gelesen und auf den Bildschirm als Rasterzeilen ausgegeben. Die Eingabe der korrigierten Daten wird über einen INPUT-Befehl abgewickelt, worauf die Daten sofort wieder in den Speicher geschrieben werden.

Zeilen 180 bis 499: In diesem Ausschnitt werden die Zeilennummern berechnet, in welchen das korrigierte Sprite abgelegt werden soll. Hierauf erfolgt die Ausgabe der neuen DATA-Zeilen und das Einlesen in den Basic-Speicher.

Zeilen 500 bis 635: Das Grundmenü wird auf den Bildschirm geschrieben und die Tastatur abgefragt.

Zeilen 900 bis 920: Dieser Block des Programms liest die DATA-Zeilen und schreibt sie in den Speicher.

(Thomas Erhart/do)

Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel {CLR}, muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksummer 64 auf Seite 135.

```
100 POKE 650,128                                <017>
102 GOSUB 900                                   <110>
105 GOTO 500                                    <065>
110 PRINT"{LIG.GREEN,CLR,RVSON,12SPACE}DAT
    ENKORREKTUR{13SPACE}"                       <159>
112 W=0:POKE 53280,11:POKE 53281,11             <079>
115 PRINT"{HOME,DOWN}"                          <197>
120 FOR ZB=0 TO 20                              <186>
130 FOR ZA=0 TO 2                               <084>
135 ZC=ZB*3+ZA:PRINT TAB(ZA*10+5);" ";          <180>
140 GG=PEEK(832+ZC):FOR X=7 TO 0 STEP-1:X$
    =CHR$(42*((GG AND 2↑X)/2↑X)+46):XX$=XX
    $+X$                                        <024>
141 PRINT X$;:NEXT:IF MC=-1 THEN PRINT TAB
    (ZA*10+5)"{8LEFT}";:FOR X=1 TO 8 STEP
    2:GOTO 155                                  <217>
142 XX$="":NEXT:PRINT"{2SPACE}":NEXT:GOTO
    165                                         <031>
144 PRINT"{HOME,DOWN}"                          <226>
145 FOR ZB=0 TO 20                              <211>
146 FOR ZA=0 TO 2                               <100>
147 ZC=ZB*3+ZA                                  <109>
148 PRINT TAB(ZA*10+4);:INPUT DA$:PRINT"{U
    P}";                                        <120>
149 DA$=LEFT$(DA$,8):FOR X=8 TO 1 STEP-1:X
    $=MID$(DA$,X,1):XX=-1*(X$="X")              <077>
150 XY=XY+XX*2↑(8-X):NEXT:IF XY<0 OR XY>25
    5 THEN 144                                  <239>
151 PRINT"{4SPACE,4LEFT}";XY;"{LEFT}";:POK
    E 832+ZC,XY                                 <229>
152 IF LEFT$(DA$,1)="£"THEN 110                 <154>
153 IF LEFT$(DA$,1)="↑"THEN 500                 <221>
154 GOTO 164                                    <050>
155 IF MID$(XX$,X,2)="XX"THEN PRINT"{RVSON
    ,YELLOW}XX";                                <082>
156 IF MID$(XX$,X,2)="X."THEN PRINT"{RVSON
    ,RED}X.";                                   <095>
157 IF MID$(XX$,X,2)=".X"THEN PRINT"{RVSON
    ,GREEN}.X";                                 <010>
158 IF MID$(XX$,X,2)=".."THEN PRINT"{RVOFF
    ,GREY 2}..";                                <102>
159 PRINT"{RVOFF,BLACK}";:NEXT:GOTO 142         <014>
164 XY=0  :NEXT:PRINT"{DOWN}";:NEXT             <164>
165 PRINT                                       <011>
```

Listing. »Sprite-Bibliothek«

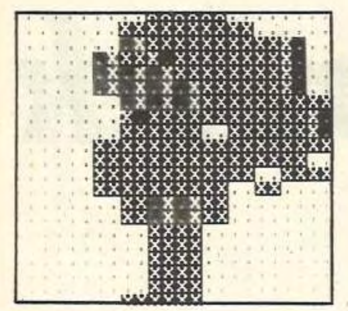




Bild 1. Mädchen

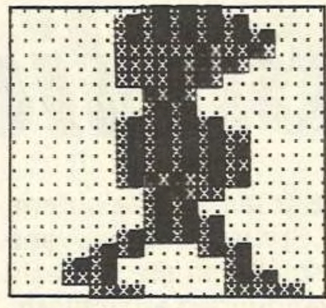


Bild 2. Mann 1

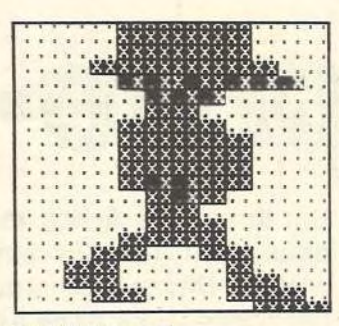


Bild 3. Mann 2

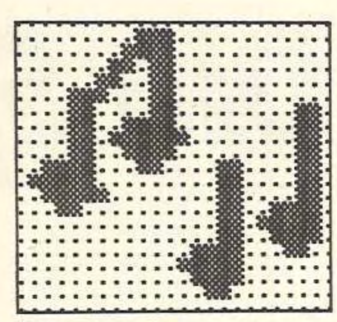


Bild 4. Note 1

```
180 PRINT"{RVSON,2SPACE}KORRIGIEREN{3SPACE
    }? (J/N) / SPEICHERN (S)";             <246>
200 GET K$:IF K$=""THEN 200                <190>
210 PRINT"{HOME,23DOWN}"                   <133>
211 PRINT"{6SPACE,RVSON,2SPACE}£ HOME{2SPA
    CE,RVOFF,7SPACE,RVSON,2SPACE}↑ MENUE{2
    SPACE,RVOFF,5SPACE}";                  <083>
220 IF K$="J"THEN W=0:GOTO 144             <236>
230 IF K$="S"THEN 260                      <036>
240 IF K$="N"THEN 500                      <136>
260 PRINT"{CLR,5DOWN}"                     <095>
300 ZN=992+SC*10                           <157>
320 FOR AA=0 TO 3                          <077>
325 PRINT ZN+AA*2;"DATA";                  <110>
330 FOR AB=0 TO 15                         <207>
340 PRINT MID$(STR$(PEEK(832+AB+AA*16))+",
    ",2,5);                                <143>
360 NEXT:PRINT"{LEFT,SPACE}":NEXT:PRINT"RU
    N"                                     <151>
380 FOR AA=0 TO 4:POKE 631+AA,13:NEXT      <153>
400 POKE 198,5                             <184>
420 PRINT"{HOME.3DOWN}"                    <094>
499 END                                    <247>
500 PRINT"{CLR,LIG.GREEN,17DOWN}"          <042>
530 PRINT"{29SPACE,RVSON}CR WEITER{SPACE,D
    OWN}"                                  <118>
531 PRINT"{RVSON}F1 BILD{RVOFF,SPACE,RVSON
    }F3 SPRITE{RVOFF,SPACE,RVSON}F5 MULTI
    1{RVOFF,SPACE,RVSON}F7 MULTI 2{DOWN}"  <176>
532 PRINT"{RVSON}F2 ART{SPACE,RVOFF,SPACE,
    RVSON}F4 X-EXPD{RVOFF,SPACE,RVSON}F6 Y
    -EXPD{SPACE,RVOFF,SPACE,RVSON}F8 AENDE
    RN"                                    <158>
540 V=53248                                <165>
560 POKE V+0 ,150:POKE V+1,100             <061>
561 POKE 2040,13                           <102>
562 POKE V+16,0  :POKE V+21,1              <240>
570 POKE 53248+28,0                        <057>
580 GET K$:IF K$=""THEN 580                <160>
590 HA=0                                   <174>
600 IF K$="{F1}" THEN HA=32                <127>
601 IF K$="{F3}" THEN HA=39:HP=8           <083>
602 IF K$="{F5}" THEN HA=37:HP=18          <158>
603 IF K$="{F7}" THEN HA=38:HP=29          <121>
605 IF HA=0 THEN 620                       <116>
610 HB=(PEEK(V+HA)AND 15)+1                <018>
611 POKE V+HA,HB                           <092>
612 IF HA=32 THEN POKE V+33,HB:GOTO 580    <129>
615 POKE 55296+HP+800,HB                   <187>
616 POKE 55297+HP+800,HB                   <060>
617 GOTO 580                               <197>
620 IF K$="{F2}" THEN HA=28:MC=NOT MC      <050>
621 IF K$="{F4}" THEN HA=29                <185>
622 IF K$="{F6}" THEN HA=23                <184>
623 IF K$="{F8}" THEN POKE V+21,0:GOTO 110 <160>
624 IF K$=CHR$(13) THEN GOSUB 900          <052>
625 IF HA=0 THEN 580                       <147>
630 HB=ABS(SGN(PEEK(V+HA))-1)              <053>
631 POKE V+HA,HB                           <112>
635 GOTO 580                               <215>
900 FOR AA=0 TO 63                         <254>
910 READ DA                                <232>
912 IF DA=999 THEN RESTORE:SC=0:GOTO 900   <219>
915 POKE 832+AA,DA                         <047>
920 NEXT:SC=SC+1:RETURN                    <128>
999 :                                      <213>
1000 BILD 1                                <217>
1002 DATA 0,63,0,0,255,192,0,127,248,1,239
     ,248,1,95,248,1                       <088>
```

```
1004 DATA 87,248,0,87,255,0,191,254,0,252,
     254,3,255,255,3,255                   <075>
1006 DATA 252,3,255,207,3,255,48,0,215,0,0
     ,215,0,0,60,0                         <100>
1008 DATA 0,60,0,0,60,0,0,60,0,0,60,0,0,25
     2,0,255                               <071>
1009 :                                     <223>
1010 BILD 2                                <035>
1011 :                                     <225>
1012 DATA 0,42,0,0,170,128,0,170,160,0,253
     ,208,0,245,64,0                       <053>
1014 DATA 245,0,0,20,0,0,40,0,0,170,0,0,17
     0,192,0,170                           <048>
1016 DATA 128,0,170,192,0,150,128,0,39,0,0
     ,40,0,0,42,0                          <226>
1018 DATA 0,170,0,2,130,128,10,0,128,14,0,
     160,3,192,252,255                     <177>
1019 :                                     <233>
1020 BILD 3                                <109>
1021 :                                     <235>
1022 DATA 0,255,192,0,255,192,0,255,192,3,
     255,240,0,91,84,0                     <114>
1024 DATA 21,0,0,60,0,0,255,0,0,255,192,0,
     255,192,0,255                         <167>
1026 DATA 192,0,215,192,0,55,0,0,60,0,0,63
     ,0,0,255,0                            <215>
1028 DATA 3,195,192,15,0,192,15,0,240,3,19
     2,252,0,0,63,255                      <054>
1029 :                                     <245>
1030 BILD 4                                <186>
1031 :                                     <247>
1032 DATA 0,0,0,0,112,0,0,240,0,1,176,0,3,
     48,0,14                               <205>
1034 DATA 48,0,12,48,6,12,240,6,13,248,6,1
     2,240,6,60,97                         <195>
1036 DATA 134,124,1,134,62,1,134,24,1,158,
     0,1,190,0,1,158                       <193>
1038 DATA 0,7,140,0,15,128,0,7,128,0,3,0,0
     ,0,0,255                              <068>
1039 :                                     <255>
1040 BILD 5                                <004>
1041 :                                     <001>
1042 DATA 0,0,0,0,0,0,14,0,0,15,0,0,13,128
     ,0,12                                 <033>
1044 DATA 192,0,12,112,0,12,48,0,60,49,128
     ,126,49,128,60,49                     <057>
1046 DATA 128,24,49,134,0,241,134,1,249,13
     4,0,241,134,0,103,134                 <155>
1048 DATA 0,15,134,0,7,30,0,3,62,0,0,30,0,
     0,12,255                              <013>
```

Listing. »Sprite-Bibliothek« (Fortsetzung)

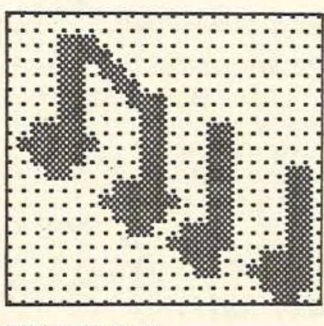


Bild 5. Note 2

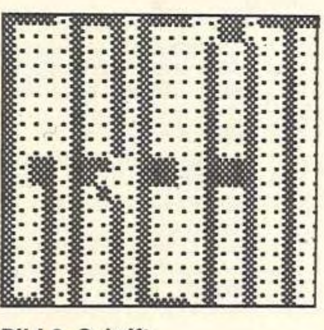


Bild 6. Schrift

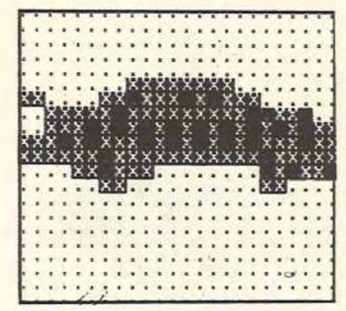



Bild 7. Automobil 1

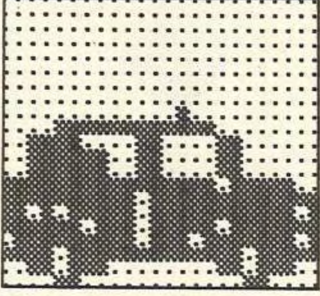


Bild 8. Automobil 2

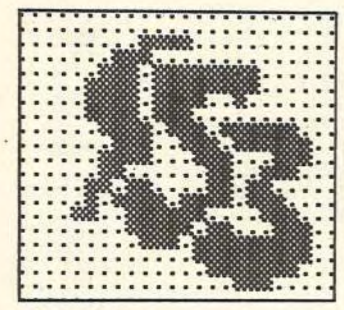


Bild 9. Ziffern

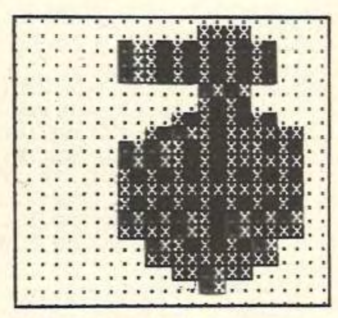
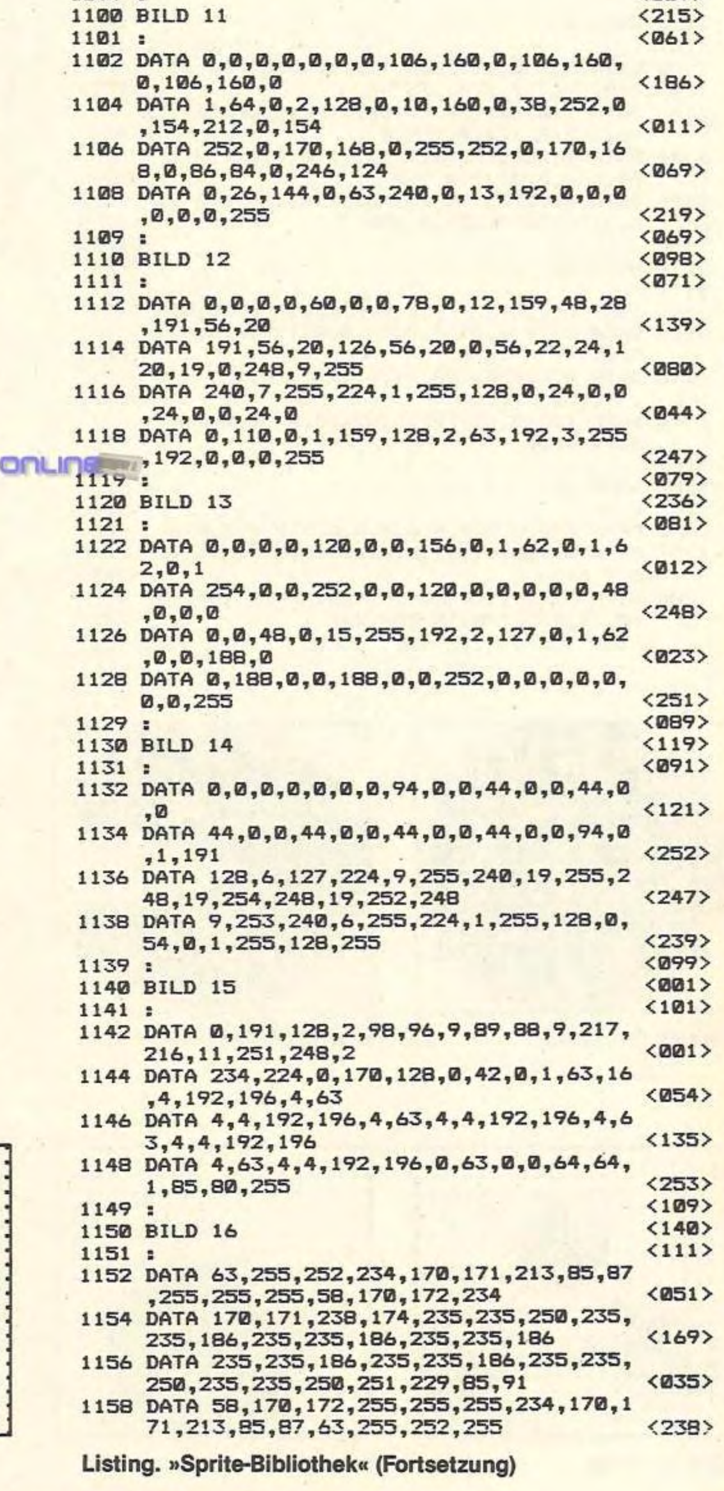


Bild 10. Bombe 1

```
1049 :                                         <009>
1050 BILD 6                                    <078>
1051 :                                         <011>
1052 DATA 247,188,223,132,160,196,132,161,
     36,132,161,36,132,161,36,132              <110>
1054 DATA 161,36,132,161,36,132,161,36,132
     ,161,36,132,161,36,183,57                 <169>
1056 DATA 228,182,57,228,149,33,36,148,161
     ,36,148,161,36,148,161,36                 <218>
1058 DATA 148,161,36,148,161,36,148,161,36
     ,148,161,36,244,189,36,255                <105>
1059 :                                         <019>
1060 BILD 7                                    <152>
1061 :                                         <021>
1062 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0      <044>
1064 DATA 255,0,195,117,192,62,255,252,58,
     170,171,235,170,187,253,255               <192>
1066 DATA 223,13,192,220,3,0,48,0,0,0,0,0,
     0,0,0,0                                   <164>
1068 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,25
     5                                         <179>
1069 :                                         <029>
1070 BILD 8                                    <227>
1071 :                                         <031>
1072 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0      <054>
1074 DATA 0,0,0,0,0,0,0,0,0,4,0,15,255,0,6
     0,48                                      <170>
1076 DATA 192,63,48,192,63,48,192,255,255,
     124,255,223,255,215,223,253               <107>
1078 DATA 253,223,127,127,221,253,247,255,
     223,55,0,220,12,0,48,255                  <151>
1079 :                                         <039>
1080 BILD 9                                    <045>
1081 :                                         <041>
1082 DATA 0,0,0,0,0,0,0,120,0,1,192,0,3,19
     1,192,7                                   <249>
1084 DATA 191,192,7,188,128,7,184,0,3,221,
     248,1,206,252,3,199                       <132>
1086 DATA 28,3,3,56,6,195,96,5,239,112,12,
     252,56,0,251,28                           <185>
1088 DATA 0,119,252,0,7,252,0,3,248,0,0,22
     4,0,0,0,255                               <033>
1089 :                                         <049>
1090 BILD 10                                   <077>
1091 :                                         <051>
1092 DATA 0,0,0,0,3,192,0,106,160,0,106,16
     0,0,106,160,0                             <164>
1094 DATA 1,64,0,2,128,0,10,160,0,38,252,0
     ,90,232,0,90                              <064>
1096 DATA 252,0,170,168,0,255,252,0,170,16
     8,0,86,84,0,246,124                       <059>
```

```
1098 DATA 0,26,144,0,63,240,0,13,192,0,1,0
     ,0,0,0,255                                <210>
1099 :                                         <059>
1100 BILD 11                                   <215>
1101 :                                         <061>
1102 DATA 0,0,0,0,0,0,0,106,160,0,106,160,
     0,106,160,0                               <186>
1104 DATA 1,64,0,2,128,0,10,160,0,38,252,0
     ,154,212,0,154                            <011>
1106 DATA 252,0,170,168,0,255,252,0,170,16
     8,0,86,84,0,246,124                       <069>
1108 DATA 0,26,144,0,63,240,0,13,192,0,0,0
     ,0,0,0,255                                <219>
1109 :                                         <069>
1110 BILD 12                                   <098>
1111 :                                         <071>
1112 DATA 0,0,0,0,60,0,0,78,0,12,159,48,28
     ,191,56,20                                <139>
1114 DATA 191,56,20,126,56,20,0,56,22,24,1
     20,19,0,248,9,255                         <080>
1116 DATA 240,7,255,224,1,255,128,0,24,0,0
     ,24,0,0,24,0                              <044>
1118 DATA 0,110,0,1,159,128,2,63,192,3,255
     ,192,0,0,0,255                            <247>
1119 :                                         <079>
1120 BILD 13                                   <236>
1121 :                                         <081>
1122 DATA 0,0,0,0,120,0,0,156,0,1,62,0,1,6
     2,0,1                                     <012>
1124 DATA 254,0,0,252,0,0,120,0,0,0,0,0,48
     ,0,0,0                                    <248>
1126 DATA 0,0,48,0,15,255,192,2,127,0,1,62
     ,0,0,188,0                                <023>
1128 DATA 0,188,0,0,188,0,0,252,0,0,0,0,0,
     0,0,255                                   <251>
1129 :                                         <089>
1130 BILD 14                                   <119>
1131 :                                         <091>
1132 DATA 0,0,0,0,0,0,0,94,0,0,44,0,0,44,0
     ,0                                        <121>
1134 DATA 44,0,0,44,0,0,44,0,0,44,0,0,94,0
     ,1,191                                    <252>
1136 DATA 128,6,127,224,9,255,240,19,255,2
     48,19,254,248,19,252,248                  <247>
1138 DATA 9,253,240,6,255,224,1,255,128,0,
     54,0,1,255,128,255                        <239>
1139 :                                         <099>
1140 BILD 15                                   <001>
1141 :                                         <101>
1142 DATA 0,191,128,2,98,96,9,89,88,9,217,
     216,11,251,248,2                          <001>
1144 DATA 234,224,0,170,128,0,42,0,1,63,16
     ,4,192,196,4,63                           <054>
1146 DATA 4,4,192,196,4,63,4,4,192,196,4,6
     3,4,4,192,196                             <135>
1148 DATA 4,63,4,4,192,196,0,63,0,0,64,64,
     1,85,80,255                               <253>
1149 :                                         <109>
1150 BILD 16                                   <140>
1151 :                                         <111>
1152 DATA 63,255,252,234,170,171,213,85,87
     ,255,255,255,58,170,172,234               <051>
1154 DATA 170,171,238,174,235,235,250,235,
     235,186,235,235,186,235,235,186           <169>
1156 DATA 235,235,186,235,235,186,235,235,
     250,235,235,250,251,229,85,91             <035>
1158 DATA 58,170,172,255,255,255,234,170,1
     71,213,85,87,63,255,252,255               <238>
```

Listing. »Sprite-Bibliothek« (Fortsetzung)

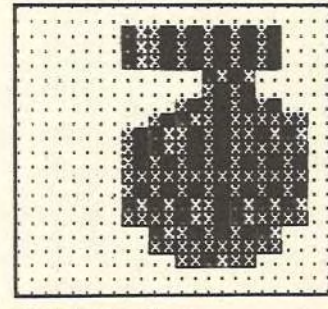


Bild 11. Bombe 2

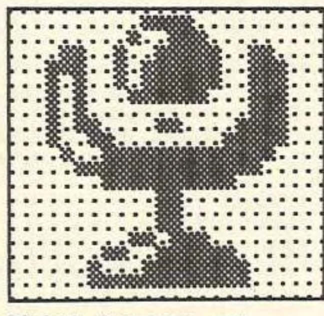
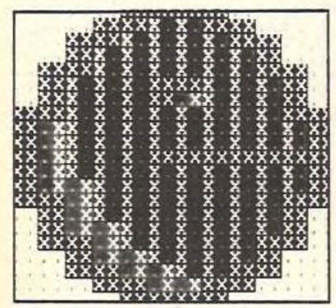
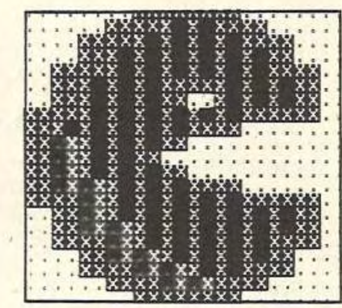
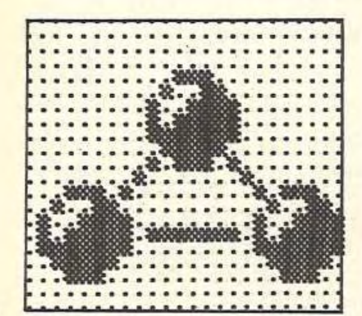


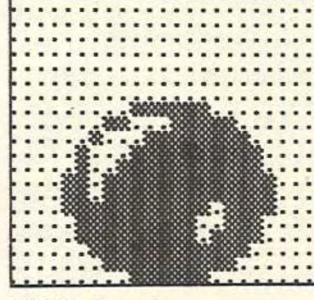
Bild 12. Gegenstand 1

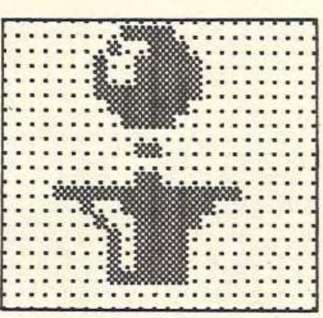
```
1159 :                                        <119>
1160 BILD 17                                  <022>
1161 :                                        <121>
1162 DATA 0,255,0,3,235,192,15,170,240,14,
     170,176,62,170,188,58                    <043>
1164 DATA 190,172,58,182,172,250,170,175,2
     18,170,171,218,170,171,218,191           <025>
1166 DATA 255,218,170,171,214,170,171,246,
     170,175,54,170,172,53,170,172            <213>
1168 DATA 61,170,188,13,106,176,15,90,240,
     3,215,192,0,255,0,255                    <152>
1169 :                                        <129>
1170 BILD 18                                  <161>
1171 :                                        <131>
1172 DATA 0,255,0,3,235,192,15,170,240,14,
     170,176,62,170,188,58                    <053>
1174 DATA 190,172,58,178,172,250,170,252,2
     34,175,0,218,176,0,218,192               <091>
1176 DATA 0,218,176,0,214,175,0,246,170,25
     2,54,170,172,53,170,172                  <248>
1178 DATA 61,170,188,13,106,176,15,90,240,
     3,215,192,0,255,0,255                    <162>
1179 :                                        <139>
1180 BILD 19                                  <043>
1181 :                                        <141>
1182 DATA 0,255,0,3,235,192,15,170,240,14,
     170,176,62,170,176,58                    <023>
1184 DATA 250,192,58,219,0,250,172,0,234,1
     76,0,218,192,0,219,0                     <056>
1186 DATA 0,218,192,0,214,176,0,246,172,0,
     54,171,0,53,170,192                      <232>
1188 DATA 61,170,176,13,106,176,15,90,240,
     3,215,192,0,255,0,255                    <132>
1189 :                                        <149>
1190 BILD 20                                  <241>
1191 :                                        <151>
1192 DATA 0,0,0,0,0,0,0,0,0,0,0,0,28,0,0     <171>
1194 DATA 38,0,0,79,0,0,95,0,0,127,0,0,62,
     0,0,93                                   <001>
1196 DATA 0,0,128,128,29,0,92,38,0,38,79,0
     ,79,95,127,95                            <076>
1198 DATA 127,0,127,62,0,62,28,0,28,0,0,0,
     0,0,0,255                                <128>
1199 :                                        <159>
1200 BILD 21                                  <123>
1201 :                                        <161>
1202 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0      <184>
1204 DATA 0,0,0,0,0,0,0,0,0,126,0,1,143,12
     8,2,63                                   <245>
1206 DATA 192,4,127,224,4,255,224,13,255,2
     40,13,255,240,15,254,240                 <089>
```

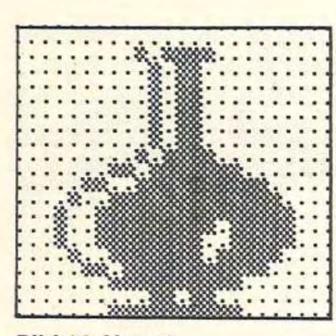

Bild 17. Packman 1

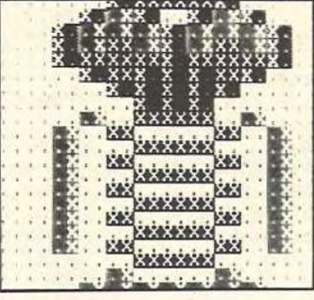

Bild 18. Packman 2

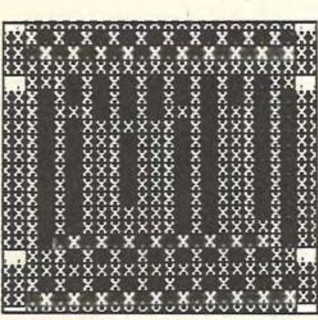

Bild 20. Atom

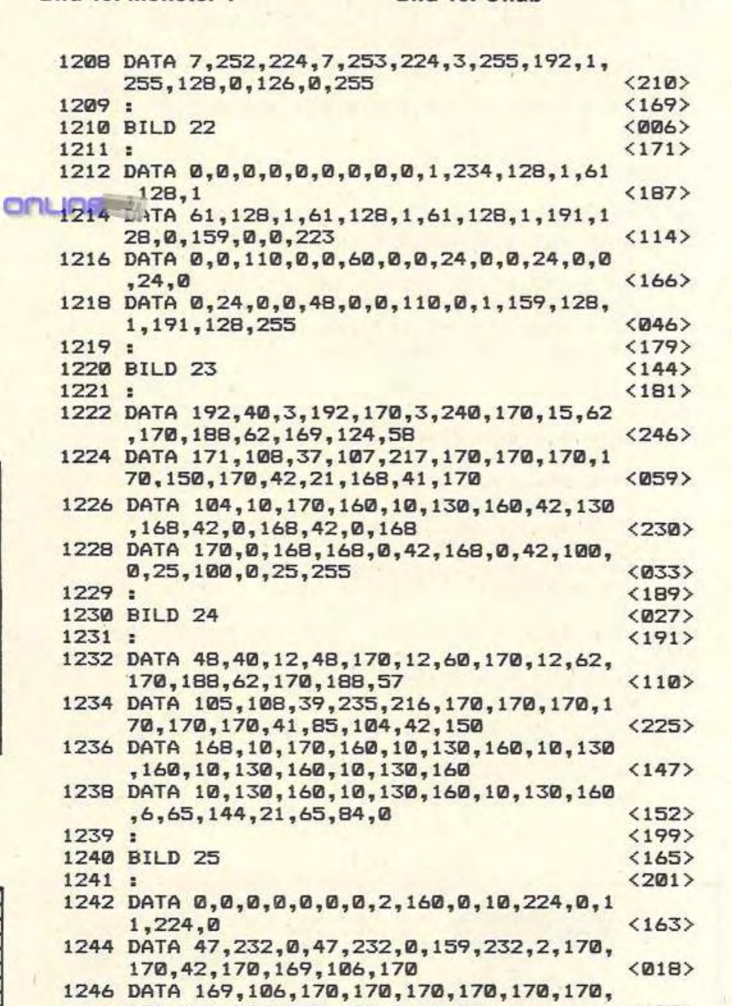

Bild 21. Kugel


Bild 13. Gegenstand 2


Bild 14. Vase 1

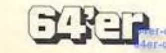

Bild 15. Monster 1

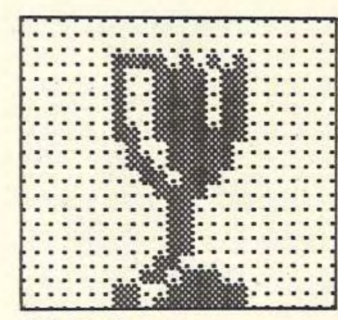

Bild 16. Ölfaß

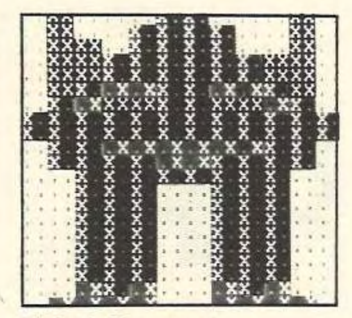


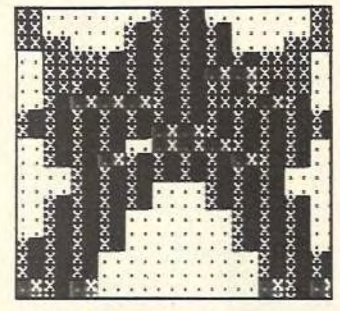
```
1208 DATA 7,252,224,7,253,224,3,255,192,1,
     255,128,0,126,0,255                      <210>
1209 :                                        <169>
1210 BILD 22                                  <006>
1211 :                                        <171>
1212 DATA 0,0,0,0,0,0,0,0,0,1,234,128,1,61
     ,128,1                                   <187>
1214 DATA 61,128,1,61,128,1,61,128,1,191,1
     28,0,159,0,0,223                         <114>
1216 DATA 0,0,110,0,0,60,0,0,24,0,0,24,0,0
     ,24,0                                    <166>
1218 DATA 0,24,0,0,48,0,0,110,0,1,159,128,
     1,191,128,255                            <046>
1219 :                                        <179>
1220 BILD 23                                  <144>
1221 :                                        <181>
1222 DATA 192,40,3,192,170,3,240,170,15,62
     ,170,188,62,169,124,58                   <246>
1224 DATA 171,108,37,107,217,170,170,170,1
     70,150,170,42,21,168,41,170              <059>
1226 DATA 104,10,170,160,10,130,160,42,130
     ,168,42,0,168,42,0,168                   <230>
1228 DATA 170,0,168,168,0,42,168,0,42,100,
     0,25,100,0,25,255                        <033>
1229 :                                        <189>
1230 BILD 24                                  <027>
1231 :                                        <191>
1232 DATA 48,40,12,48,170,12,60,170,12,62,
     170,188,62,170,188,57                    <110>
1234 DATA 105,108,39,235,216,170,170,170,1
     70,170,170,41,85,104,42,150              <225>
1236 DATA 168,10,170,160,10,130,160,10,130
     ,160,10,130,160,10,130,160               <147>
1238 DATA 10,130,160,10,130,160,10,130,160
     ,6,65,144,21,65,84,0                     <152>
1239 :                                        <199>
1240 BILD 25                                  <165>
1241 :                                        <201>
1242 DATA 0,0,0,0,0,0,0,2,160,0,10,224,0,1
     1,224,0                                  <163>
1244 DATA 47,232,0,47,232,0,159,232,2,170,
     170,42,170,169,106,170                   <018>
1246 DATA 169,106,170,170,170,170,170,170,
     170,171,253,255,220,5,64,84              <150>
1248 DATA 5,64,84,1,0,16,0,0,0,0,0,0,0,0
     ,255                                     <249>
1249 :                                        <209>
1250 BILD 26                                  <048>
1251 :                                        <211>
1252 DATA 0,0,0,0,0,0,84,40,0,93,85,80,245
     ,85,84,85                                <145>
```

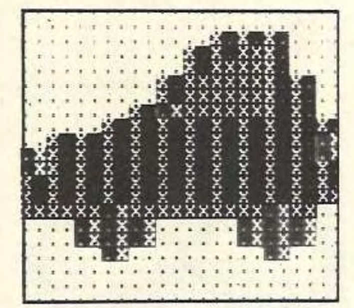
Listing. »Sprite-Bibliothek« (Fortsetzung)




Bild 22. Vase 2

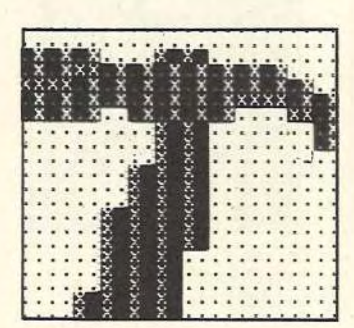

Bild 23. Monster 2

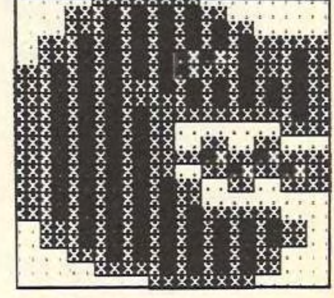

Bild 24. Monster 3

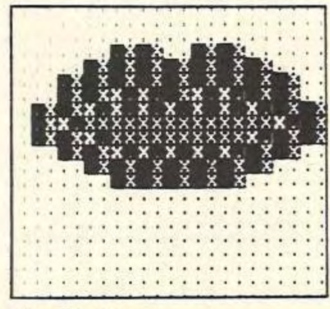

Bild 25. Automobil 3

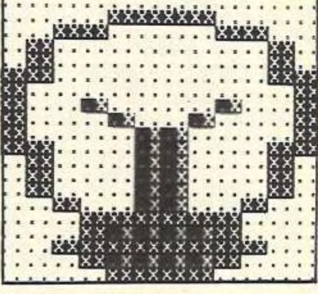
```
1254 DATA 85,245,84,85,13,0,40,1,0,40,1,0,
     40,0,0,168                            <068>
1256 DATA 0,0,168,0,0,168,0,2,168,0,2,168,
     0,2,168,0                             <137>
1258 DATA 2,160,0,2,160,0,2,160,0,10,160,0
     ,10,160,0,255                         <161>
1259 :                                     <219>
1260 BILD 27                               <186>
1261 :                                     <221>
1262 DATA 3,252,0,14,171,0,14,170,192,58,1
     70,191,58,165,187,234                 <114>
1264 DATA 167,187,234,234,171,234,250,171,
     234,191,251,234,176,15,234,179        <221>
1266 DATA 48,234,189,223,234,179,119,234,1
     72,207,234,172,0,234,171,240          <030>
1268 DATA 58,170,172,58,170,172,14,170,176
     ,3,234,176,0,63,192,255               <132>
1269 :                                     <229>
1270 BILD 28                               <069>
1271 :                                     <231>
1272 DATA 0,0,0,0,0,0,0,0,0,2,138,0,10,170
     ,128,42                               <061>
1274 DATA 170,160,41,101,160,165,85,104,15
     9,255,216,167,255,104,41,85           <213>
1276 DATA 160,10,170,128,2,170,0,0,0,0,0,0
     ,0,0,0,0                              <091>
1278 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,25
     5                                     <133>
1279 :                                     <239>
1280 BILD 29                               <209>
1281 :                                     <243>
1282 DATA 0,0,0,0,0,0,0,255,0,15,0,240,48,
     0,12,48                               <034>
1284 DATA 0,12,240,0,15,192,0,3,193,0,67,1
     92,65,3,192,20                        <155>
1286 DATA 3,240,20,15,48,20,12,48,20,12,48
     ,20,12,12,20,48                       <246>
1288 DATA 3,255,192,3,85,192,15,255,240,3,
     85,192,0,255,0,255                    <194>
1289 :                                     <251>
1290 BILD 30                               <151>
1291 :                                     <253>
1292 DATA 0,0,0,0,0,0,0,0,0,15,2,128,55,20
     1,160,55                              <222>
1294 DATA 201,160,63,202,160,15,202,128,0,
     200,0,0,32,0,0,32                     <049>
1296 DATA 0,0,32,0,0,140,0,10,143,192,38,1
     41,240,38,141,240                     <025>
1298 DATA 42,143,240,10,3,192,0,0,0,0,0,0,
     0,0,0,255                             <221>
1299 :                                     <005>
1300 BILD 31                               <033>
1301 :                                     <007>
1302 DATA 0,0,0,0,0,0,0,0,0,0,60,0,0,23,0,
     0                                     <000>
1304 DATA 7,0,0,21,0,0,21,0,0,4,0,0,247,19
     2,1,255                               <000>
1306 DATA 208,5,63,20,6,63,36,0,110,64,0,1
     06,64,0,34,0                          <234>
1308 DATA 0,34,0,0,34,0,0,34,0,0,179,128,0
     ,243,192,255                          <228>
1309 :                                     <015>
1310 BILD 32                               <172>
1311 :                                     <017>
1312 DATA 0,60,0,0,251,0,0,255,0,2,235,0,1
     4,235,0,62                            <116>
1314 DATA 183,0,58,191,176,250,174,176,251
     ,186,240,250,171,240,242,167          <141>
1316 DATA 192,194,165,128,67,170,192,66,25
     5,128,156,170,128,14,162,128          <218>
1318 DATA 14,162,160,12,160,160,0,0,160,0,
     0,240,0,0,60,255                      <097>
1319 :                                     <025>
1320 BILD 33                               <054>
1321 :                                     <027>
1322 DATA 0,28,0,0,127,128,0,63,192,0,63,1
     92,0,23,128,0                         <073>
1324 DATA 127,128,0,31,0,0,15,0,0,29,128,0
     ,3,192,0,7                            <020>
1326 DATA 192,3,159,192,3,247,128,3,135,12
     8,0,15,0,0,7,128                      <174>
1328 DATA 0,15,128,0,29,128,0,184,192,0,11
     2,192,0,35,192,255                    <006>
1329 :                                     <035>
1330 BILD 34                               <193>
1331 :                                     <037>
1332 DATA 0,28,0,0,127,128,0,191,192,0,63,
     192,0,23,128,0                        <063>
1334 DATA 63,128,0,31,0,0,15,0,0,29,128,0,
     3,192,0,7                             <202>
1336 DATA 192,0,15,192,0,31,128,0,7,128,0,
     15,0,0,7,128                          <081>
1338 DATA 0,15,128,0,29,128,0,184,192,0,11
     2,192,0,35,192,255                    <016>
1339 :                                     <045>
1340 BILD 35                               <075>
1341 :                                     <047>
1342 DATA 0,171,127,252,72,248,30,133,240,
     15,3,224,3,131,192,3                  <047>
1344 DATA 131,192,1,135,192,0,199,128,1,71
     ,128,3,103,128,6,63                   <219>
1346 DATA 0,7,255,0,7,255,0,13,127,128,13,
```

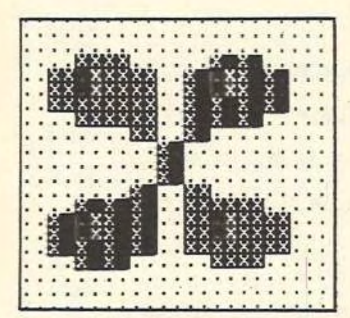
Listing. »Sprite-Bibliothek« (Fortsetzung)

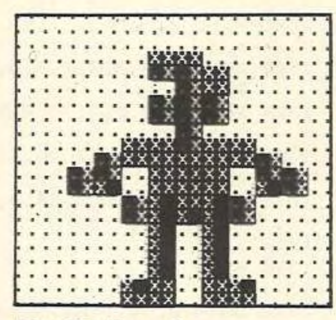

Bild 26. Hammer

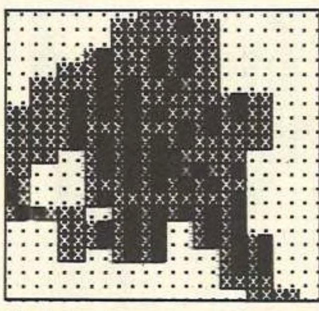

Bild 27. Monster 4

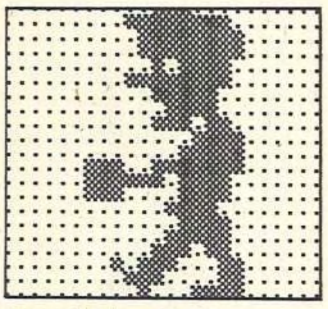

Bild 28. Lippen


Bild 29. Glühbirne


Bild 30. Gegenstand


Bild 31. Mann 3

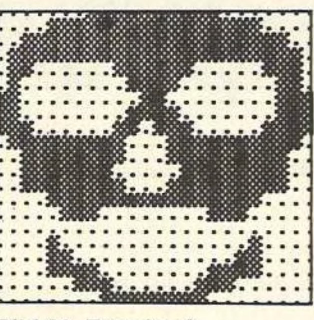

Bild 32. Mann 4

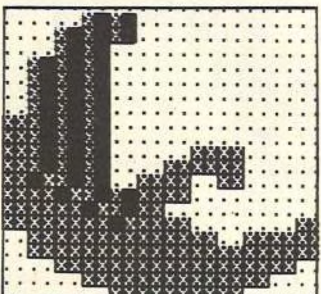

Bild 33. Mann 5

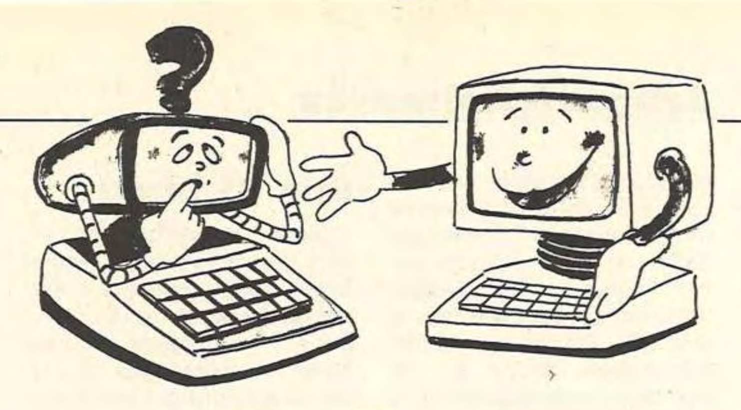
```
     127,128,9,63,128                         <162>
1348 DATA 9,61,128,0,24,192,0,8,64,0,28,22
     4,0,20,160,255                           <195>
1349 :                                        <055>
1350 BILD 36                                  <214>
1351 :                                        <057>
1352 DATA 1,255,128,15,255,240,63,255,252,
     127,255,254,112,126,6,96                 <126>
1354 DATA 60,2,224,24,3,224,60,3,240,118,7
     ,127,227,254,63,227                      <002>

1356 DATA 252,63,193,252,63,227,252,15,255
     ,240,7,0,224,0,0,0                       <152>
1358 DATA 8,0,16,12,0,48,15,0,240,3,129,19
     2,0,255,0,255                            <230>
1359 :                                        <065>
1360 BILD 37                                  <096>
1361 :                                        <067>
1362 DATA 0,0,0,2,128,0,10,128,0,10,0,0,42
     ,0,0,42                                  <207>
1364 DATA 0,0,42,0,0,42,0,0,234,0,0,234,0,
     0,234,3                                  <198>
1366 DATA 192,234,15,192,218,60,192,246,18
     8,0,253,176,0,255,124,3                  <187>
1368 DATA 63,255,63,63,255,255,15,255,252,
     3,255,240,0,255,192,255                  <111>
1369 :                                        <075>
9999 DATA 999                                 <238>
```

Listing. »Sprite-Bibliothek« (Schluß)


Bild 36. Totenkopf


Bild 37. Banane



Fragen & Antworten

Der PET-Emulator der Demodiskette

Was hat es mit dem Programm »PET-Emulator« auf der C64-Demodiskette auf sich?

WOLFGANG JOACHIM

Ohne Frage läßt die Dokumentation von Commodore auch zur Demodiskette einiges zu wünschen übrig. Das Programm »Emulator« läßt sich jedoch trotzdem sehr gut verwenden. Es gibt inzwischen zwar schon eine große Auswahl an Programmen für den C64, aber einige Probleme (zum Beispiel mathematischer Natur) sind noch nicht umgesetzt worden. Da bietet sich doch eine Übernahme der alten CBM 3032-Programme an.

Konstruktionsbedingt unterscheiden sich der 3032 und der C64 aber stark in der Speicherbelegung, dem Speicherplatz und der Zeropage. Zum Beispiel liegt das Video-RAM des 3032 im Bereich von 32768 bis 33792. Der C64 dagegen hat sein Video-RAM ab Adresse 1024 bis 2023 und benötigt zusätzlich noch die Information über die Zeichenfarbe. Der 3032 verfügt über 31 KByte freies RAM, der C64 dagegen über 38 KByte.

Um nun 3032-Programme direkt und ohne Umschreibung auf dem C64 laufen zu lassen, braucht man nur den Emulator von der Demodiskette laden und starten. Danach wird das betreffende 3032-Programm von Kassette oder Diskette geladen und ist ohne weiteres sofort lauffähig. Das gilt sowohl für Basic- als auch für reine Maschinenprogramme.

Bei einigen wenigen Maschinenprogrammen kann der Emulator aussteigen, beispielsweise wenn auf ROM-Routinen zugegriffen wird, die der Emulator nicht unterstützt. Da hilft nur eines: ausprobieren.

Der Emulator enthält übrigens einen DOS-Manager, der dem DOS 5.1 der Demodiskette entspricht.

Insgesamt ist der Emulator eine nützliche Sache. Er vergrößert die ohnehin schon bemerkenswerte Programmvielfalt für den C64 noch um einiges.

STEFAN ULLMANN

Probleme mit Datasette

Ich habe andauernd »LOAD ERROR« bei meiner Datasette.

PETER RITTER

Bei Problemen mit Datasette hilft es oft, den Tonkopf mit Spiritus zu putzen (Wattestäbchen verwenden). Außerdem sollte man keine Chromdioxid-Kassetten verwenden. Auch die Justierung des Tonkopfes mittels der dafür gedachten Schraube bringt manchmal Erfolg.

KLAUS KOHLER

Drucker zu langsam?

Die an meinen C64 angeschlossene Typenrad-Schreibmaschine »Privileg 3000«, betrieben mit Textomat von Data Becker ist mir viel zu langsam. Wer kann helfen? Und wie?

JÜRGEN HEESCH

Typenraddrucker sind leider nun einmal um einiges langsamer als Matrix- oder Tintenstrahldrucker. Dieser Mangel ist konstruktionsbedingt und läßt sich softwaremäßig nicht beheben. Es ist ähnlich wie mit den Autos der 18-PS-Klasse – sie sind einfach von Natur aus langsam. Die einzige Abhilfe besteht in der Anschaffung eines schnelleren Modells oder auch eines Matrixdruckers mit NLQ-Schrift.

Reset-Schalter zerstört Computer?

Ich wollte einen Reset-Schalter in meinen C64 einbauen lassen. Mein Händler sagt aber nein dazu. Frage: Stimmt es, daß nach einem Reset durch Kurzschluß einige Bausteine im Computer zerstört werden können?

LOTHAR BIRKENSTOCK

Sie sind einem Ammenmärchen aufgesessen. Ein ordnungsgemäß eingebauter Reset-Schalter kann nichts zerstören. Der Netzschalter Ihres Computers ist beispielsweise ebenfalls mit der RESET-Leitung des Prozessors verbunden, und niemand würde auf die Idee kommen zu behaupten, das Einschalten eines Computers könnte diesen zerstören.

Simons Basic und DOS 5.1

Wie kann man gleichzeitig mit Simons Basic und DOS 5.1 arbeiten?

DETLEV PREISLER

Beide Programme arbeiten wie folgt zusammen:
- Simons Basic laden und starten.
- DOS 5.1 mit "LOAD" "DOS5.1",8,1 laden und mit SYS 52224 starten.

Es geht natürlich auch umgekehrt, kann aber dann in einigen Fällen zu Problemen führen.

Hier nun die Abhilfen:
- Sind beide Programme im Speicher, dann wird DOS mit SYS 52224 eingeschaltet.
- Werden DOS-Befehle nur mit einem müden READY beantwortet (oder überhaupt nicht), dann gibt man " #8" ein. Die »8« steht dabei für die Geräteadresse der Floppy.

Es gibt aber auch eine Simons Basic Version II, die von Anfang an problemlos mit dem DOS 5.1 läuft.

Basic-Zeilen länger als 80 Zeichen?

Die maximale Zeilenlänge beim C64 beträgt 80 Zeichen. Bei manchen Programmen werden diese teilweise überschritten. Wie kann man das Programm trotzdem zum Laufen bringen?

RAINER JOST

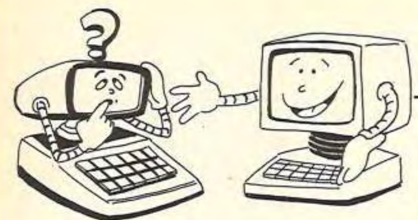
Diese häufig gestellte Frage läßt sich einfach beantworten. Im Handbuch des C64 befindet sich im Anhang D auf Seite 130/131 eine Tabelle der Abkürzung von Basic-Schlüsselwörtern. Fast jeder Basic-Befehl läßt sich durch Eingabe des ersten Buchstabens, gefolgt vom geshifteten zweiten Buchstaben des Befehls abkürzen. Zum Beispiel wird aus der Eingabe von P (Shift)O das Wort POKE, oder P (Shift)R wird zu PRINT# im Gegensatz zu »?« für PRINT. So kann eine Basic-Zeile beim Listen 80 Zeichen überschreiten. Versuchen Sie nicht, diese Zeile hinterher zu verändern, indem Sie mit dem Cursor in diese Zeile gehen und nach einer Änderung wieder mit RETURN verlassen. Dann wird automatisch der letzte Teil, der sich in der dritten Reihe befindet, anbgeschnitten. Zum Editieren geben Sie am besten die komplette Zeile mit den entsprechenden Abkürzungen noch einmal ein.

Nochmals Simons Basic

Benötigt Simons Basic einen Teil des Basic-RAMs? Wenn ja, wieviel?

LARS STEUBESAND

Simons Basic belegt die oberen acht KByte des Anwender-RAM, der verfügbare freie Speicher für Basic-Programme reduziert sich also um diese acht KByte.



Grafik mit 1526?

Ich besitze den 1526-Drukker von Commodore. Wie kann man ihn grafikfähig machen?
PERCY DAHM

Der Commodore ist hardwaremäßig nicht für hochauflösende Grafik vorgesehen, das heißt, es besteht keine Möglichkeit einer Einzelnadelansteuerung. Allerdings kann man ein Zeichen selbst definieren und so – allerdings sehr umständlich – Grafik drucken. Dazu definiert man sich das Zeichen als geeignetes Grafiksymbol, fährt durch Ausgabe einer Reihe von Blanks an die richtige Position, druckt das Zeichen und veranlaßt dann einen Wagenrücklauf ohne Zeilenvorschub. Nun wird das nächste Grafikzeichen definiert und das ganze Spiel wiederholt sich. Mit dieser Methode ist sogar eine Grafik-Hardcopy möglich (siehe 64'er, 5/84, Seite 74: »Ein eigentlich unmögliches Programm«).

Rechengenauigkeit

Warum gibt der C64 bei »PRINT INT(3/0.03)« nicht 100, sondern 99 aus?

Mathematisch gesehen ist die INT-Funktion eine Abbildung der Menge der reellen Zahlen auf die Menge der ganzen Zahlen, mit der Eigenschaft: INT(x) ist diejenige ganze Zahl, die kleiner oder gleich (!) x ist. Anders ausgedrückt schneidet INT einfach alle Nachkommastellen einer Zahl ab. Und genau das macht auch der C64.

Warum erhält man dann aber die falsche Antwort »INT(3/0.03) = 99«?. Dazu muß ich etwas ausholen. Testen Sie mal das folgende Programm:

```
10 K = 0 : FOR I = 0 TO 2 STEP 0.1
20 IF I = 1 THEN PRINT "I = 1
GEFUNDEN!" : K = 1
30 NEXT
40 IF K = 0 THEN PRINT "I
= 1 NICHT GEFUNDEN"
50 END
```

Wenn Sie keinen Tippfehler gemacht haben und keine Sonderversion des C64 besitzen, erhalten Sie als Ausgabe »I = 1 NICHT GEFUNDEN!«. Erstaunlich! Wie kann so etwas sein? Wenn man von Null bis Zwei in Zehntel-Schritten zählt, erreicht man doch ganz sicher auch die Eins. Nun, der C64 scheinbar nicht. Er stellt nämlich intern alle Zahlen binär dar, also auch solche kleiner als Eins. Fatalerweise ist es nun so, daß dezimal endliche Brüche in Binärschreibweise häufig nur als nicht abbrechende, periodische Zahlen geschrieben werden können. Das ist speziell bei 0.1 und auch bei 0.03 der Fall. Also wird die interne Binärdarstellung dieser Zahlen im Computer nach einer gewissen Stellenzahl abgebrochen (gerundet). Darum gibt der Ausdruck »3/0.03« beim Computer nicht ganz exakt 100, sondern etwas weniger.

Die Differenz können Sie sich mit »PRINT 100 - 3/0.03« selbst anschauen.

Wendet man nun die INT-Funktion auf eine Zahl an, die etwas kleiner ist als 100, dann kann nur 99 dabei herauskommen.

Das auf den ersten Blick merkwürdige Ergebnis »INT (3/0.01) = 99« ist also nicht auf die INT-Funktion zurückzuführen (diese arbeitet völlig einwandfrei), sondern ist eine Folge der internen Binärdarstellung von Dezimalzahlen im C64.

Aus diesem Grunde ist bei »IF...THEN«-Abfragen auch Vorsicht geboten. So manches selbstgeschriebene Programm arbeitet in manchen Situationen scheinbar nicht korrekt, weil einfach nicht an die nur begrenzte Rechengenauigkeit eines Computers gedacht wurde.
JÜRGEN BUSCH

RENEW mit einem POKE?

Mit dem POKE 2050,10 bekommt man Programme, die mit SYS 64738, mit RESET oder mit NEW gelöscht wurden, wieder! Nach der Eingabe einer nicht belegten Programmzeile und (RETURN) ist das Programm wieder da.
OLIVER BECKER

Schön wär's ja, aber leider nicht ganz richtig. Zwar kann man das Programm mit diesem POKE wieder sichtbar machen, doch lauffähig ist es nicht mehr. Sobald in dem Programm eine Variable definiert wird, hat man nur noch Schrott vor sich, auch Speichern nach dem »RENEW« hilft nichts mehr. Der POKE 2050,10 setzt nämlich die Pointer nicht mehr neu.

Geht TI$ falsch?

Die eingebaute Uhr (TI$) verliert die richtige Zeit, wenn Daten oder Programme auf Kassette gespeichert werden. Nach Beendigung eines Abspeichervorganges hat die Uhr einen Sprung nach vorne getan, der jedoch leider keinerlei Beziehung zur tatsächlich vergangenen Zeit hat. Das ist bei meinem Programm-Thermometer mit stündlichem Abspeichern der Temperatur auf Kassette sehr störend.
ROLF TULEWSKI

Sowohl die interne Uhr (TI$) als auch die Kassettenoperationen werden beim VC 20 und C64 über Interrupt gesteuert. Da es beim Laden und Speichern von Programmen und Daten auf hohe zeitliche Genauigkeit ankommt, wird die interne Uhr für die Dauer dieser Operationen einfach abgeschaltet. Nach Beendigung der Kassettenopertion kann TI$ daher einen nicht definierten Wert besitzen. Leider gibt es keine Möglichkeit, dies zu umgehen, es sei denn, man verwendet einen externen Zeitgeber.

Zeichensalat?

Es kommt sehr oft vor, daß ich nach dem LISTen eines Programms nur noch merkwürdige Zeichen und Zeilen voller unsinniger Basic-Befehle sehe, wie zum Beispiel »10 PRINTPRINTSAVELISTGET...« etc. Das Ganze wird zudem noch in vielen verschiedenen Farben angezeigt, teilweise auch in reverser Darstellung. Ich kann diese Zeichen dann auch nicht überschreiben. Woran liegt das, wie bekomme ich das weg?
DIRK SCHEPANEK

Der beschriebene Effekt kann drei verschiedene Ursachen haben:

1. Sie haben ein Maschinenprogramm ohne Basic-Vorspann, das mit ,8,1 geladen werden muß, nur mit ,8 geladen.

2. Sie haben ein Basic-Programm, das den unteren Basic-Bereich Speicherplatz für Grafik benötigt, einfach geladen, ohne mit bestimmten POKE-Befehlen den Basic-Start hochzusetzen.

3. Sie haben durch mehr oder weniger wildes POKEn in von Basic oder Betriebssystem benutzten Speicherstellen den Computer völlig irritiert.

Es gibt nun Gott sei Dank einen bemerkenswert einfachen Weg, diesen »Zeichensalat« wieder zu beseitigen: durch Aus- und Wiedereinschalten des Computers.

Eingaben von der Tastatur verhindern?

Wie verhindere ich Eingaben von der Tastatur (insbesondere die STOP-Taste)? Kann man das mit einem POKE-Befehl verhindern?
KLEMENS HÖGENAUER

Eingaben von der Tastatur verhindert man mit POKE 649,0 (in Speicherstelle 649 ist die Größe des Tastatur-Puffers gespeichert).

Die RUN/STOP- und RESTORE-Taste sperrt man durch POKE 808,227. Die Sperre wird gelöst durch POKE 649,10:POKE 808,237.

Text und Grafik mischen

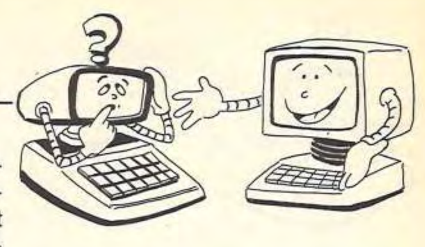
Wie kann ich in Simons Basic hochauflösende Grafik und Text gleichzeitig darstellen?
FRANK SEGER

Die Darstellung von Text und gleichzeitig Grafik im Grafikmodus des Simons Basic läßt sich durch den Simons Basic-Befehl »TEXT« erreichen. Dieser ist so anzuwenden:

TEXT x,y,"Text",m,gr,ab
x = X-Koordinate;
y = Y-Koordinate;
m = Grafikmodus
(m=0 oder m=1 oder m=2),
gr = Größe des darzustellenden Buchstabens;
ab = Abstand zwischen den einzelnen Buchstaben.

Um einen gleichen Abstand und eine gleiche Größe der Buchstaben wie beim normalen Textmodus zu erreichen, wird für »gr« eine 1 und für »ab« eine 7 eingesetzt. Also:
TEXT x,y,"Text",m,1,7

Dennoch läßt sich eine Eingabe von Antworten im Grafikmodus des Simons Basic nicht sehr einfach realisieren, da bei der Eingabe, jedenfalls im Grafikmodus, weder Cursor noch Eingaben sichtbar sind. Vielleicht gibt es einen Ersatzbefehl für den herkömmlichen INPUT?
STEPHAN HARDY



Programmunterbrechung bei Druckerausgabe

Bei der Druckerausgabe erhalte ich oft einen »DEVICE NOT PRESENT ERROR«, obwohl der Drucker eingeschaltet ist.

RUDOLF OTT

Dieser Fehler tritt sehr häufig bei Druckern auf, die über den seriellen Bus des C64 angeschlossen werden. Die Ursache dafür liegt in der Konzeption des seriellen Busses. Abhilfe: Statt PRINT# mit CMD arbeiten, dann tritt der Fehler nur noch äußerst selten oder gar nicht mehr auf. MARC HABER

Schwierigkeiten mit Softwareschutz?

Wieso ist das Programm auf einer geschützten Diskette nach einem »Backup« nicht lauffähig?

CARONNI GERMANO

Weil der Software-Hersteller durch den Kopierschutz gerade ein »Backup« verhindern will. Zu diesem Zweck sind meist gleich mehrere Sicherungen eingebaut. Um es mit den Software-Herstellern nicht zu verderben, wollen wir Ihnen aber lieber nicht verraten, wie das im einzelnen funktioniert.

Starten und Speichern von Spielprogrammen beim Commodore 64

Wenn ich ein Spielprogramm von Kassette geladen habe, starte ich es normalerweise durch die Eingabe des Wortes RUN nebst Drücken der RETURN-Taste. In einigen Fällen funktioniert das nicht, aber ich habe bisher folgende Verfahren kennengelernt:

1. Eingabe von SYS mit Adreßzahl, wobei die Adreßzahl im meist nur einzeiligen LIST-Ausdruck steht.

2. Eingabe von SYS 64738 und dann erst SYS mit der Adreßzahl aus dem Listing.

3. Eingabe von SYS mit Adreßzahl, die nicht im Listing steht (wie kann ich diese finden, wenn ich sie nicht kenne oder vergessen habe?)

4. Löschen der Programmzeilen mit SYS-Angaben.

5. Drücken der RUN/STOP- und RESTORE-Taste.

Mich interessiert, wie diese Startroutinen zu erklären sind und ob sie bei einem neuen Speichern des Programmes zu eliminieren sind. Häufig kann man aus einem Spielprogramm nicht in den Normalzustand zurückkehren. Mir bleibt dann nichts anderes übrig, als den Commodore ab- und wieder einzuschalten.

In manchen Fällen erscheint nach Drücken von RUN/STOP und RESTORE auf leerem Bildschirm das Wort READY und darunter der blinkende Cursor, aber nach Eingabe von Befehlen wie LOAD oder RUN wird nur die Eingabe von Befehlen wie LOAD oder RUN gelöscht, und der Cursor springt zurück. Was bedeutet das? Wie kann ich hier den Commodore wieder zu sinnvoller Reaktion veranlassen?

Manchmal passiert mir folgendes: Ich möchte ein Spielprogramm auf Kassette speichern. Obwohl das Programm mit Namen auf der Kassette ist, kann ich es mit diesem Namen nicht SAVEn. Der Commodore meldet einen Fehler. Lasse ich den Namen weg, funktioniert das Speichern einwandfrei. Wie ist das zu erklären und wie kann ich in solchen Fällen den Namen doch mitspeichern?

WERNER KIEFERT

Zu den Fragen zum Starten von Programmnamen:

1. Hier handelt es sich um ein Maschinensprache-Programm, an das vorher eine Basic-Zeile gehängt wurde, die mit dem entsprechenden SYS-Befehl den Maschinenspracheteil startet. (Der Speicher des C64 ist aufgeteilt in die Speicherzellen 0 bis 65535. In einer dieser Speicherzellen liegt der Beginn des Maschinenprogramms. Mit dem SYS-Befehl und der Angabe der Speicherzelle, wo das Maschinenprogramm beginnt, wird es gestartet).

2. Wenn man erst mit SYS 64738 den Computer in den Einschaltzustand bringt und dann den entsprechenden SYS-Befehl aus dem Listing eingibt, müßte eigentlich das gleiche herauskommen, wie wenn man den SYS 64738 wegläßt. Hier hat Ihnen jemand Humbug erzählt oder Ihnen ein ziemlich verpfuschtes Programm gegeben.

3. Hier wurde ein Maschinensprache-Programm abgespeichert, ohne eine Basic-Zeile mit dem entsprechenden SYS-Befehl vorne dranzuhängen. Die Adresse für den Start des Programms läßt sich am besten so finden:
- Maschinensprache lernen
- Maschinensprache-Monitor laden
- SUCHEN

4. Ist uns nicht bekannt.

5. Der sogenannte Restore-Vektor wurde vom Programmierer geändert, das heißt, bei Drücken von RUN/STOP-RESTORE wird nicht mehr die normale Betriebssystem-Routine gestartet, sondern es wird zu der Adresse gesprungen, die der Programmierer angewählt hat, in diesem Fall die Startadresse des Programms.

Zur Frage, die das Zurückkehren aus Spielprogrammen in den Normalmodus betrifft:

Das war offensichtlich Absicht der Programmierer der Spiele (aus welchem Grund auch immer). Hier hilft meistens wirklich nur das Aus- und Anschalten des Computers.

Antwort auf die Frage, die das Speichern von Programmen betrifft:

Hier reicht offensichtlich der Speicher nicht mehr aus, um den Namen dort abzulegen. (Der Fehler ist dann wohl ein OUT OF MEMORY ERROR.) Man kann sich jedoch manchmal helfen, indem man den Variablenspeicher in einen freien Speicherbereich verlegt:

```
LOAD »Programm«
POKE 56,207
CLR
SAVE »Name«
```

Für das Funktionieren dieser Maßnahme kann nicht garantiert werden, da es doch immer sehr auf das einzelne Programm ankommt.

Bits hörbar machen

Ist es beim C64 möglich, beim Laden von Datasette die Signale hörbar zu machen?

WERNER FRINGS

Um die Signale, die von der Datasette kommen, hörbar zu machen, muß einfach mit POKE54296,15 die Lautstärke im SID auf 15 gesetzt werden. Nun sind die Signale beim Laden, SAVEn und VERIFYen über den Lautsprecher des Monitors zu hören (auch bei Fast- und Turbo-Tape).

STEFAN GOSSENS

Die Zwangspausen des C64

Bei längerem Suchen in einer indexsequentiellen Datei macht der C64 Pausen bis zu zirka fünf Minuten (bei älteren Versionen von Datamat bis zu 20 Minuten!). Woher kommen die Pausen und wie sind sie zu beseitigen?

HENRY KÖNIG

Wenn der C64 sehr viele Strings zu verarbeiten hat, muß er ab und zu seinen dafür benötigten Speicherplatz aufräumen. Das heißt, er überprüft, welche Variablen noch aktuell sind und benötigt werden. Den Rest schmeißt er weg. Die dadurch entstandenen Lücken werden durch die verbleibenden Werte geschlossen. Diese Vergleiche und Verschiebearbeiten sind der Grund für diese langen Wartezeiten. Den Vorgang selber nennt man auch Garbage Collection. Mehr zu diesem Thema finden Sie in der Ausgabe 1/85.

Wie schließt man Monitore an?

Frage: Wie schließe ich diverse Monitore an den C64 oder VC 20 an?

(verschiedene Fragensteller)

Einen Schwarz-Grün-Monitor schließt man an, indem man aus dem Video-Port des C64 die Drähte »GND« und »VIDEO HIGH« mit den entsprechenden Monitoreingängen verbindet. Dabei muß »GND« am Monitor-Stecker außen liegen.

ANDREAS ROESCHIES

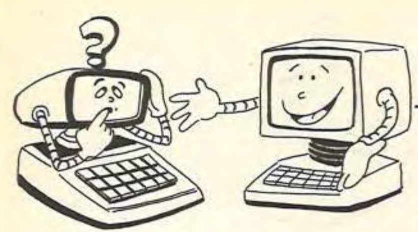
C64 und Videorecorder

Wie kann ich mit dem Videorecorder die Signale vom Computer aufnehmen?

EUGEN ANGER

Die Lösung ist ganz einfach: Man nehme das Antennenkabel des Computers, stecke dieses in den Antenneneingang des Videorecorders und stellte den Recorder auf den »Sender« des Computers ein. Danach braucht man nur noch wie bei der normalen Aufnahme zu verfahren.

STEFAN WÖSSNER





Hardcopies mit MPS 802 und Simons Basic?

Wir haben den C64 und den Drucker MPS 802 (Nachfolgegerät vom VC 1526), dazu Simons Basic. Ist es möglich, im HiRes- beziehungsweise Lowcol-Modus eine Kopie des Bildschirms zu erstellen?

GERHARD SCHIEF

Das könnten Sie ja eigentlich selbst ausprobieren. Aber wenn Sie mit dem Befehl HRDCPY keine Hardcopy des normalauflösenden Bildschirms erhalten sollten, könnte der Drucker defekt sein. Eine Hardcopy des hochauflösenden Grafikbildschirms hingegen ist mit dem Simons Basic-Befehl COPY nicht möglich. Das gilt auch für den weitgehend baugleichen VC 1526. Bei HiRes-Hardcopies macht der MPS 802 Schwierigkeiten, da er von sich aus nicht grafikfähig ist. Erst durch geeignete Programme läßt er sich dazu »überreden«.

Erhöhte Lebensdauer bei Dauerbetrieb?

Wie lange kann man den C64 laufen lassen, ohne daß er Schäden nimmt?

BURCHARD LAASCH

Es gibt Computer-Benutzer, die ihren Computer niemals abschalten. Er läuft dort jahrelang, ohne daß deswegen ein Defekt auftritt. Sie begründen das damit, daß bei jedem Ein- und Ausschaltvorgang die elektronischen Bauelemente warm werden und wieder abkühlen. Da in vielen Bauelementen verschiedene Materialien verarbeitet werden, die sich bei einer Temperaturänderung unterschiedlich ausdehnen, können Spannungen auftreten. Diese andauernden thermischen Belastungen könnten die Lebensdauer vor allem der empfindlichen Halbleiter herabsetzen.

Bei einem Dauerbetrieb treten diese Belastungen nicht auf. Wenn für ausreichende Belüftung gesorgt wird und der Computer nicht noch zusätzlich auf einer Heizung abgestellt wird und vor direkter Sonneneinstrahlung geschützt ist, dürften keine Probleme entstehen.

Resettaste für VC 20?

Sind die Anschlüsse für eine Reset-Taste am User-Port von VC 20 identisch? Ist so eine Taste auch am C64 anschließbar?

ADRIAN SCHNEIDER

Die beiden User-Ports sind nicht 100% identisch, jedoch gibt es beim Anschluß eines Reset-Tasters keine Unterschiede. Durch Verbindung von Pin 1 und Pin 3 erfolgt bei beiden Computern ein Reset.

Raubkopien strafbar?

Darf man gekaufte Software auf Leerkassetten kopieren und verkaufen? Oder ist das gesetzlich verboten, so daß eine Bestrafung erfolgt?

MAIK ROGALLA

Das Kopieren und Weiterverkaufen von fremder Software ist ein Verstoß gegen das Urheberrecht und wird entsprechend geahndet. Man muß in jedem Falle mit hohen Schadensersatzforderungen der Herstellerfirmen rechnen.

Programme weg beim Reset?

Ich habe in meinem C64 eine Reset-Taste eingebaut, und wenn ich diese benutze, ist das Programm weg. Wie kann ich dieses Programm wieder hereinholen? Suche dazu die PEEKs und POKEs mit Adressen.

ARNDT WÖRAU

Nach einem Reset bleiben Maschinenprogramme nach wie vor im Speicher erhalten und können wieder mit entsprechendem SYS gestartet werden. (Dies ist jedoch nicht der Fall bei Maschinenspracheprogrammen, die im Kassettenpuffer liegen [dez. 828-1019], da dieser bei einem Reset gelöscht wird.) Bei Basic-Programmen muß man schon mehr machen, um sie wieder herzuholen:

```
POKE 2049,99: POKE 2050,99:
POKE 45,X: POKE 46,Y: SYS
42291
```

Dabei sind X und Y die Werte, die man vor Drücken des Reset-Knopfes durch ?PEEK(45); PEEK(46) abfragen sollte. Kommt man aus Versehen auf den Knopf und weiß den Inhalt dieser Speicherstellen nicht mehr, so empfiehlt es sich, in die Speicherstellen 45 und 46 angenommene Werte für die Programmenadresse zu POKEn (45 = Low-Byte, 46 = High-Byte).

Von Kassette auf Floppy?

Ich möchte mir demnächst eine Floppy 1541 kaufen. Kann man mehrere Programme, die man auf Kassette hat (gekaufte Spiele und anderes), auf eine einzige Diskette speichern, um Platz zu sparen? Wenn ja, wie?

ANDREAS SCHMELZER

Natürlich kann man auch mehrere Programme auf einer Diskette abspeichern. Kommerzielle Software auf Kassette ist jedoch in der Regel mit Autostart und einem Softwareschutz versehen, so daß ein Kopieren in der Regel nicht möglich ist.

Druckroutinen umschreiben?

Wie kann ich Programme, die für den Drucker VC 1526 geschrieben sind und dessen Fähigkeiten ausnutzen, für den VC 1525 umschreiben?

BRUNO SCHIFFER

Der VC 1526 verfügt über eine Reihe komfortabler Befehle zur formatierten Ausgabe, die auf dem VC 1525 nicht vorhanden sind. Diese Formatierungsbefehle müssen für den VC 1525 durch Software ersetzt werden. Im Prinzip geht es darum, eine Art PRINT USING-Routine zu schreiben. Am besten schreibt man alle Programmteile, die Ausgaben an den Drucker liefern, völlig neu.

POKEs im Simons Basic?

Warum bringen verschiedene POKEs, die im Commodore-Basic einwandfrei funktionieren (zum Beispiel Listschutz) mit Simons Basic das Programm zum Abstürzen?

SEBASTIAN OBERMAYR

Einige Adressen der Zeropage (das sind die Speicherzellen von 0 bis 255) werden von Simons Basic anders belegt, so daß eine Reihe von POKE-Befehlen nicht mehr funktionieren oder eben zum Absturz führen.

Disketten beidseitig verwenden?

Wenn man mit einem Bürolocher auf einer einseitig beschreibbaren Diskette am linken Rand eine Stanzung in Höhe des Schreibschutzes am rechten Rand vornimmt, läßt sich dann die Diskette beidseitig auf einer Floppy-Disk 1541 verwenden? Ist die Datensicherheit dabei gewährleistet, oder spricht etwas gegen diese Methode?

DIPL.-ING. M. LOHSE

Die Floppy VC 1541 interessiert sich nicht für das kleine Indexloch in der Diskette, wie andere Floppy-Stationen das in der Regel tun. Deshalb können Disketten beidseitig verwendet werden. Allerdings ist bei einseitig beschreibbaren Disketten auch nur eine Seite durch den Hersteller geprüft. Die zweite Seite kann daher unter Umständen fehlerhafte Sektoren enthalten, die aber vom Floppy-Betriebssystem dann nicht benutzt werden.

Für sehr wichtige Aufzeichnungen empfiehlt sich jedoch immer die Verwendung einseitig geprüfter Disketten, die auf der Rückseite nicht beschrieben werden.

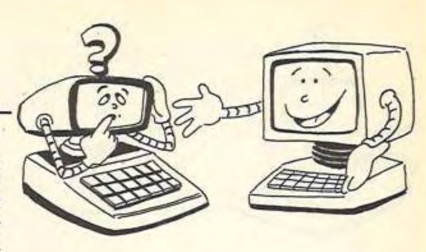
RESET über User-Port?

Um bei meinem Commodore 64 nach einem Spiel ein anderes Programm zu laden, muß in den meisten Fällen der Computer aus- und dann wieder eingeschaltet werden. Kann man das umgehen, indem man kurzzeitig Pin 1 und 3 des User-Ports miteinander verbindet, oder führt dies zu einer Beschädigung des Computers?

WALTER STILLER

Pin 1 des User-Ports liegt an Masse, Pin 3 ist die RESET-Leitung. Stellt man kurzfristig eine Überbrückung her (zum Beispiel mit einer Büroklammer oder ähnlichem), dann wird ein Reset ausgelöst, wie der Computer ihn auch nach dem Einschalten durchführt.

Der Reset an sich ist ungefährlich. Falls man jedoch beim Manipulieren am User-Port unabsichtlich andere Pins miteinander in Verbindung bringt, kann dies zu Beschädigungen des CIA-Bausteins führen. Es empfiehlt sich daher der Einsatz eines kleinen Tastschalters.



Commodore-64-Programme auf VC 20?

Sind die für den C64 abgedruckten Programme auch für den VC 20 zu benutzen? Natürlich nur, soweit die benötigte Peripherie vorhanden ist.

ANDREAS VOLZ

Sofern es sich um reine Basic-Programme ohne POKEs und PEEKs handelt, gibt es keine Schwierigkeiten bei der Übernahme von Programmen des C64 auf den VC 20 oder umgekehrt. Wegen der unterschiedlichen Zeichenzahl pro Bildschirmzeile (22 Zeichen beim VC 20, 40 Zeichen beim C64) wird die Bildschirmdarstellung unter Umständen jedoch etwas unübersichtlich. Entsprechende Änderungen bei den PRINT-Befehlen sind jedoch auch für den Anfänger nicht allzu schwer zu bewerkstelligen. Falls das Programm aber POKE-Befehle oder Maschinenspracheteile enthält, ist die Anpassung ohne Kenntnis der jeweils anderen Systemadressen kaum möglich. Mehr zu diesem Thema finden Sie im Sonderheft 3/86.

Drucker/Floppy-Einschalt-Test

Mit welchen Befehlen kann ich überprüfen, ob der Drucker oder die Floppy eingeschaltet ist?

ERNST JESCHKE

Eine Überprüfung, ob ein Drucker am seriellen Bus angeschlossen ist, läßt sich beim Commodore 64 durch folgende Befehle erreichen:

```
100 POKE 768, 185
110 OPEN 1,4: PRINT#1
120 POKE 768, 139
```

Da so jegliche Fehlermeldung unterdrückt wird, kann nun durch Abfrage der Statusvariablen »ST« geprüft werden, ob die Leerzeile (oder jeder beliebige andere Text, wie zum Beispiel nicht druckende Steuerzeichen!) vom Drucker angenommen wurde (IF ST < > 0 THEN ...). Somit kann verhindert werden, daß ein Programmlauf durch »DEVICE NOT PRESENT« abgebrochen wird. Ein Test auf Vorhandensein einer Diskettenstation kann durch entsprechende Programmänderung erfolgen.

RALPH BABEL

Autostart

Wie bringe ich meine Programme dazu, daß sie nach dem Laden von Diskette oder Kassette automatisch starten?

CARSTEN BRUCH

Beim Laden von Datasette ist ein Autostart sehr einfach: Drücken Sie gleichzeitig »SHIFT« und die »RUN/STOP«-Taste. Es wird das nächste Programm von Kassette gelesen und sofort gestartet. Beim Diskettenbetrieb empfiehlt es sich, zunächst das DOS 5.1 von der Demo-Diskette zu laden. Der DOS-Befehl »↑« (Hochpfeil), gefolgt von einem Programmnamen, lädt das entsprechende Programm von Diskette und startet es automatisch. Mehr zu diesem Thema finden Sie im 64'er, Ausgabe 6, 7 und 8/84.

Komma als Satzzeichen

Kann man bei einem INPUT-Befehl das Komma als normales Satzzeichen verwenden?

GERHARD GIESSMANN

Um einen INPUT-Befehl auch Kommata (und Doppelpunkte) akzeptieren zu lassen, muß man lediglich als erstes Zeichen ein Anführungszeichen eingeben, das nicht in die Variable nach »INPUT« übernommen wird. Diese Eingabe läßt sich auch automatisch über den Tastaturpuffer simulieren, indem man »POKE 631,34:POKE 198,1« vor den »INPUT« setzt. Jetzt wird das Anführungszeichen von selbst ausgegeben. Eleganter ist es natürlich, wenn eine eigene Eingaberoutine verwendet wird, wie beispielsweise »INPUTFORM« und »INPUTLINE« im EXBASIC LEVEL II.

RALPH BABEL

Datasette oder Diskette?

In welchem Umfang kann – ich bin Anfänger – die Datasette ein Diskettenlaufwerk ersetzen?

HANS GEORG WALTHER

Beide sind Massenspeicher – wem das Diskettenlaufwerk zu teuer ist, kann (und muß) ein Kassettenlaufwerk, also die Datasette, nehmen. Das Kassettenlaufwerk ist deutlich langsamer als das Diskettenlaufwerk. Dazu kommt ein zweiter Nachteil: Auf dem Magnetband in der Kassette werden die Daten sequentiell, also der Reihe nach hintereinander gespeichert. Im Falle einer Dateiverwaltung kommt also erst beispielsweise die Adresse von Herbert Adam, dann die von Erich Berger und so weiter. Wenn Sie die Adressen in der Reihenfolge brauchen, in der sie gespeichert sind, funktioniert das ganz gut; wenn Sie – bei alphabetischer Reihenfolge – erst die Adresse von Weber, dann die von Berger, dann die von Müller und so weiter brauchen, muß das Band immer erst zu der entsprechenden Stelle vor- beziehungsweise zurückspulen, was äußerst zeitaufwendig ist. Außerdem macht es einige Arbeit, in eine gegebene Reihenfolge später neue Adressen einzufügen. Würde als Speichermedium eine Diskette verwendet, so könnte auf jede Adresse (oder jeden anderen Datensatz) direkt zugegriffen werden (sogenannter Random Access), ohne daß ein Vor- oder Rücklauf des Bandes abgewartet werden muß.

Außerdem müssen Sie auf der Diskette die Daten nicht unbedingt in einer ganz bestimmten Reihenfolge speichern (Details sind von der Dateiorganisation und damit der Software abhängig). Faustregel: Mit einem Diskettenlaufwerk arbeitet man schneller und bequemer als mit einem Kassettenlaufwerk. Alle Anwendungen, die sich mit einem Kassettenlaufwerk realisieren lassen, sind auch mit einem Diskettenlaufwerk zu verwirklichen, während es umgekehrt eine Reihe von Anwendungen gibt, die nur mit einem Diskettenlaufwerk sinnvoll zu bewältigen sind.

Schreibmaschine als Drucker?

Welche Schreibmaschine verfügt über eine Schnittstelle für den VC 20?

HARALD KÜST

Prinzipiell lassen sich alle elektrischen Schreibmaschinen, die überhaupt für den Anschluß an einen Computer vorgesehen sind, auch am VC 20 betreiben. In den meisten Fällen ist dazu ein V.24-Interface notwendig, einige Schreibmaschinen lassen sich jedoch auch direkt über den seriellen Bus des VC 20 anschließen, zum Beispiel Brother EP-20C oder Olympia Compact 2. Sie müssen sich allerdings darüber im klaren sein, daß eine Schreibmaschine weder Grafik- noch Steuerzeichen des VC 20 drucken kann.

Commodore-64-Programme in den C16 laden?

Wie kann man Programme, die mit dem C64 auf Kassette gespeichert wurden, in den C16/116 laden?

WERNER MARZILIUS

Uns ist leider keine Möglichkeit bekannt, C64- oder VC 20-Software mittels Datasette in den C16 zu laden.

Der einzige Ausweg besteht im Umweg über ein Floppy-Laufwerk.

VC 20-Simulator gesucht

Ich interessiere mich sehr für einen VC 20-Simulator auf dem C64. Existiert so ein Programm überhaupt?

HELLMUT KORNDÖRFER

Nein, so etwas gibt es bestimmt nicht. Der VC 20 ist sowieso im Basic identisch zum C64, und ein Simulator kann in der Regel nur das Basic simulieren, nicht aber die völlig unterschiedliche Hardware.

Kleinschrift auf VC 1515

Ich besitze einen VC 20 mit Drucker VC 1515. Wie kann ich ein Programm-Listing im Kleinschriftmodus abdrucken?

MARTIN QUINKE

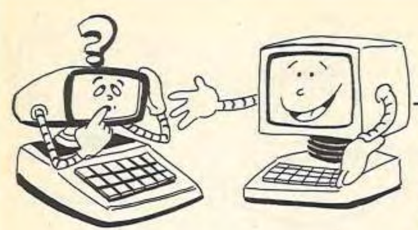
Der VC 1515-Drucker wird mit folgender Kommandozeile zur Ausgabe eines Listings im Kleinschriftmodus veranlaßt:

```
OPEN 4,4,7:PRINT#4,CHR$(17):CMD 4:LIST.
```

Vorsicht, die Angabe der Sekundäradresse 7 nicht vergessen! Mit

```
OPEN 4,4:PRINT#4,CHR$(145)«
```

kann man wieder in den Normalmodus zurückschalten (nachzulesen im VC 1515-Handbuch Seite 30/31.





Wie ist das mit Copyright?

Woher bekommt man das Copyright für seine Programme? Bringt es irgendwelche Vorteile, wenn man es hat?

HELLMUT KORNDÖRFER

Das Copyright besitzt man automatisch für jedes Programm, das man selbst geschrieben hat. Ein Copyright-Vermerk in einem Programm (oder in einem Buch) dient nur dazu, anderen Personen anzuzeigen, bei wem die Rechte für das Produkt liegen, und daß man das Programm (oder Buch) nicht einfach kopieren und weiterverkaufen darf. Eigene Programme dürfen Sie selbstverständlich verkaufen.

Leichter und besser als Basic?

Welche zweite Programmiersprache, die leichter zu erlernen und leistungsstärker als Basic ist, würden Sie mir empfehlen?

MARKUS HAUTMANN

Eine derartige Computersprache werden Sie kaum finden. Es ist ähnlich wie mit der Frage nach einem Sportwagen, der größer, schneller und besser als ein Porsche ist, aber auf jeden Fall viel weniger kosten soll. Beide Faktoren, Leistungsstärke und leichte Erlernbarkeit, schließen sich in der Regel aus. An sich ist Basic die am leichtesten zu erlernende Programmiersprache. Einfache Grafikprogrammierung erreichen Sie mit Comal (sehr empfehlenswert, da auf Basic aufbauend) oder mit Logo. Mit Pascal läßt sich besonders »elegant« programmieren, es ist allerdings nicht ganz einfach zu lernen. Wenn es Ihnen auf Geschwindigkeit ankommt, sollten Sie sich vielleicht einmal mit Assembler oder Forth beschäftigen. Oftmals tut allerdings auch schon eine Basic-Erweiterung oder ein Compiler gute Dienste. Bei der Auswahl einer Programmiersprache kommt es halt immer darauf an, was man programmieren möchte. Dazu müßten Sie sich zuerst mal darüber klar werden, was Ihnen im Basic fehlt.

Assembler und andere Mysterien

Ich habe drei Fragen an Sie:
1. Was ist ein Assembler?
2. Was ist ein Disassembler?
3. Was ist ein Maschinensprache-Monitor?
Was kann man damit machen, und wie funktionieren sie?

MICHAEL PITZ

Wie Sie vielleicht wissen, versteht der im C64 arbeitende 6510-Prozessor Basic nicht direkt, sondern nur eine viel einfacher (für die Maschine) aufgebaute »Sprache«, eben die sogenannte Maschinensprache. Sie können mit Ihrem Computer nur deshalb in Basic arbeiten, weil der C64 ein Programm, selbst in dieser Maschinensprache geschrieben, fest gespeichert hat, das einen »Basic-Prozessor« darstellt (genauso wie Sie in Basic ein Programm schreiben können, das zum Beispiel eine Dateiverwaltung darstellt).

Aber natürlich können Sie Ihren C64 auch in Maschinensprache direkt programmieren – das ist quasi seine Muttersprache. Allerdings geht das von Basic aus nicht ganz so einfach, genauso wie Sie aus einem laufenden Dateiverwaltungsprogramm heraus schlecht in Basic programmieren können. Daher gibt es spezielle Hilfsprogramme, die das ermöglichen. In der einfachsten Form handelt es sich dabei um Maschinensprache-Monitore, mit denen man – vereinfacht gesagt – kleine Maschinenprogramme gleich in der dem Prozessor geläufigen Zahlenform eingeben kann.

Dieses Programmieren in Zahlen ist zwar für die Maschine sehr angenehm, aber weniger für den Menschen. Daher wurden zu jedem Maschinenbefehl sogenannte »Mnemonics«, also leicht merkbare Abkürzung für die Wirkungsweise des Befehls, geschaffen. Um diese Mnemonics wiederum in das der Maschine verständliche Zahlenformat zu übersetzen, gibt es wiederum andere Programme – die Assembler.

Ein Disassembler wiederum tut das Gegenteil vom Assembler: mit ihm können fertige Maschinenprogramme wieder in die Assembler-Mnemonics zurückübersetzt werden. Wenn Sie sich für Maschinensprache oder Assembler interessieren, empfehlen wir Ihnen unbedingt das 64'er-Sonderheft 8/85 oder den Kurs »Von Basic zu Assembler« im 64'er-Magazin.

Hardware-Sorgen

1. Kann ein Schaden an den Geräten (Zentraleinheit, Floppy und Farbfernseher) entstehen, wenn ich sie alle drei auf einmal einschalte oder muß ich erst den Fernseher einschalten, warten bis ein Bild da ist, dann die Zentraleinheit in Betrieb setzen und nachher das Floppy-Laufwerk einschalten?

2. Was kann an der Floppy-Station und der Diskette kaputtgehen, wenn sich eine solche während des Ein- und Ausschaltens im Laufwerk befindet? Mir ist das schon öfter passiert, aber soweit ist noch alles ganz.

3. Kann es sein, daß es den C64 mit verschiedenen Betriebssystemen gibt? Mein Freund hat ein Spiel »Popeye«, das bei ihm einwandfrei läuft. Wenn wir es aber bei mir spielen wollen, macht es immer einen Aussteiger.

4. Was kann an der Hardware kaputtgehen, wenn ich während des Betriebs ein Verbindungskabel löse?

MICHAEL REICHEL

1. Den Geräten ist es völlig egal, in welcher Reihenfolge sie eingeschaltet werden. Besonders bequeme Menschen, wie beispielsweise die 64'er-Redakteure, lösen das Einschaltproblem besonders elegant: Alle Geräte werden an eine Steckdosenleiste mit Schalter angeschlossen. Liegt diese auch noch auf dem Boden, läßt sich mit einem einfachen Fußdruck die gesamte beschriebene Hardware (auch Drucker) einschalten. Vorsicht ist allerdings bei älteren Fernsehgeräten geboten. Diese vertragen es teilweise nicht, wenn die Netzspannung direkt und nicht über den eingebauten Schalter gesteuert wird.

2. Am Floppy-Disk-Laufwerk wird normalerweise kein Schaden entstehen. Disketten können aber theoretisch durch die plötzliche Spannungsspitze im Gerät teilweise gelöscht werden. Uns ist das bisher auch noch nicht passiert, trotzdem sollte hier Vorsicht geboten sein.

3. Es gibt verschiedene Versionen des C 64. Die Unterschiede sind allerdings minimal und treten nur vereinzelt zu Tage. So gibt es mehrere Ausführungen der CPU, was sich bei einigen eigentlich nicht vorgesehenen Maschinenbefehlen bemerkbar macht (Illegal Opcodes). Auch die Bus-, Sound- und Videobausteine werden in verschiedenen, aber fast identischen Versionen gefertigt.

4. Das sollte man tunlichst unterlassen. Gerade die Busbausteine in der 1541-Floppy sind da sehr empfindlich. Besonders gefährlich sind hier statische Aufladungen. Wer beim Umstecken seine Floppy zerstört, weil er in geladenem Zustand die Ports berührt, der darf mit Reparaturkosten bis zu 100 Mark und mehr rechnen. Das gleiche gilt für Computer (Serieller Bus, Datasetten-Anschluß, Userport und sogar Joystickanschluß) und Drucker.

Von Basic zur Maschinensprache?

Ist es möglich, ein Basic-Programm mit Hilfe eines Maschinenprogramms in ein Maschinenprogramm zu verwandeln? Wenn ja, womit?

FRANK WILD

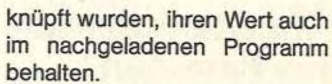
Solche Programme gibt es tatsächlich. Sie heißen »Basic-Compiler«. Ein solcher Compiler übersetzt ein Basic-Programm in Maschinensprache, die der Computer direkt verstehen und ausführen kann. Derart übersetzte Programme sind daher um einiges schneller als entsprechende reine Basic-Programme. Einen ausführlichen Test verschiedener Basic-Compiler für den C64 finden Sie in den 64'er-Ausgaben 2/85 und 4/85.

Was ist der MSE?

Ich verstehe den Aufbau der MSE-Sprache nicht. Warum sind in einer Zeile jeweils nur neun verschiedene Befehle?

TIM WENSKY

»MSE« ist keine eigene Computersprache, sondern nur ein Eingabesystem für Programme in Maschinensprache. Diese werden in einer besonderen Form als »Hexadezimalzahlen« abgedruckt. Dieses Zahlensystem ist nicht wie die uns geläufigen Dezimalzahlen auf der Basis 10 aufgebaut, sondern



auf der Basis 16. Als zusätzliche Ziffern werden dabei einfach die ersten Buchstaben des Alphabets genommen. Statt dezimal...8,9,10,11,12,13,14,15,16,... wird hexadezimal folgendermaßen gezählt:...8,9,A,B,C,D,E,F, 10,...

Jeweils zwei Hexadezimalziffern bilden ein »Byte«, also eine Zahl zwischen 0 und 255 (hexadezimal 0 und FF). Ein Byte entspricht also dem Inhalt einer Speicherzelle. In einer MSE-Zeile stehen jeweils acht Programm-Bytes. Die letzte Zahl ist eine Prüfsumme. Sie wird verwendet, um Fehleingaben in einer Zeile weitgehend auszuschließen.

Bitte lesen Sie beim Abtippen von MSE-Programmen die zugehörige Bedienungsanleitung besonders aufmerksam durch. Versuchen Sie insbesondere nie, mit dem MSE eingegebene Programme einfach mit LOAD zu laden und mit RUN zu starten. Vielmehr müssen Sie MSE-Programme (wie alle Maschinenprogramme) mit »LOAD "name",8,1« laden und in der Regel mit SYS starten.

63229 Byte free?

Ich habe den Tip bekommen, daß die Befehlsfolge »POKE 56,255 : SYS 58234« beim VC 20 einen freien Speicherplatz von über 61000 Bytes ergeben soll. Ich habe nun diese Befehlsfolge auf meinem C 64 eingegeben und erhalte tatsächlich nach der Abfrage »PRINT FRE(0) + 216« die Antwort »63229«.

Sind diese 63229 Bytes wirklich frei zum Programmieren und wird dadurch auch keine andere Funktion beeinflußt?

TORSTEN NIEK

Na, da hat Sie aber wirklich einer an der Nase herumgeführt. Der POKE-Befehl setzt zwar (allerdings nur rein rechnerisch) die Speichergrenze für den Basic-Interpreter nach oben, aber sonst tut sich überhaupt nichts. Sie haben weiterhin Ihre knappen 38 KByte Basic-Speicher, weil Sie mit einem POKE-Befehl schwerlich Hardware-Aufbau und Betriebssystem eines Computers ändern können. Im Gegenteil, wenn Sie versuchen mit Variablen zu arbeiten, kann das sogar zu einem Absturz des Computers führen, weil dieser verzweifelt versucht, Variableninhalte im ROM abzulegen. Dies kann nachtürlich nicht gutgehen.

Der SYS-Befehl hinter dem POKE ist übrigens reine Augenwischerei, denn der bewirkt nur einen Sprung in die NMI-Routine, also dasselbe, was beim Drücken der Tastenkombination RUN/STOP-RESTORE passiert.

Ärger mit der 1541

Ich habe einige Schwierigkeiten mit meinem Floppy-Laufwerk. Wenn es längere Zeit in Betrieb und ziemlich warm geworden ist, nimmt die Diskette kein Programm mehr an. Beim Speichern meldet die Floppy dann Fehler 20 oder 27. Das Laufwerk lädt dann auch nicht mehr, auch keine Directories von anderen Disketten. Nach längerem Abkühlen kann wieder geladen werden, aber nicht mehr die Programme, die ich vorher abspeichern wollte. Auf dem Laufwerk ist noch Garantie und es war auch schon zur Reparatur beim Händler. Der sagte, es sei eine Neueinstellung vorgenommen worden. Aber der Fehler tritt immer wieder auf. Schadet es der Floppy, wenn ich einen Reset auslöse?

JOSEF SPIERTZ

Ihr Problem beruht auf der thermischen Ausdehnung von Metallen. Die Mechanik des Laufwerks dehnt sich normalerweise nur minimal aus. Bei Ihnen scheint der unglückliche Extremfall vorzuliegen, daß durch diese Ausdehnung der Schreib-/Lesekopf total verstellt wird. Dann kann die Floppy-Station weder Daten lesen noch schreiben. So, wie bei Ihnen geschildert, darf dieses Problem aber unter normalen Umständen nicht auftreten. Normalerweise ist ein mehrstündiger Betrieb, solange nicht laufend formatiert wird, ohne Probleme möglich. Es gibt hier zwei Lösungsmöglichkeiten: Entweder Sie sorgen für Kühlung, indem Sie die Floppy »offen«, das heißt ohne Gehäusedeckel betreiben oder einen Lüfter verwenden. Die Alternative wäre, daß Sie auf einen Umtausch des Laufwerks bei Ihrem Händler bestehen.

Bei einem Reset am Computer wird auch in der Floppy ein Reset ausgelöst. Das hat keinerlei schädliche Folgen für Floppy und Diskette.

C 16-Sprites?

Ich besitze einen C 16 mit Datasette und wollte wissen, ob ich Sprites im Textmodus simulieren kann oder ob jemand ein Programm dafür geschrieben hat.

FRANK KARNER

Der C 16 besitzt die Shape- beziehungsweise Sprite-Fähigkeit nur im HiRes-Modus, weil im TED (der Chip, der am C 16 auch für die Video-Ausgabe zuständig ist) Sprites nicht hardwaremäßig vorgesehen sind wie beim Videochip des C 64.

Falls man aus Speicherplatzgründen auf HiRes-Shapes verzichten will oder muß, kann man sich immer noch, ähnlich wie beim VC 20, mit einem selbstdefinierten Zeichensatz helfen, der nur 2 KByte vom RAM abknappst und dennoch schon beachtliche Grafikmöglichkeiten bietet. Im Übrigen finden Sie in unserem C 16-Sonderheft (3/86) auch einen Grundlagenkurs für Grafik.

Unsichtbar nachladen?

Wie schaffe ich es, in Basic ein Nachlade-Programm zu konzipieren, das bei entsprechender Menüwahl ein Teilprogramm nachlädt – und zwar ohne Ausgabe der Meldungen »Searching for« und »Loading«?

IPEC CÜNEYT

Beim Nachladen von Programmen müssen Sie folgendes speziell beachten: Der Befehl »LOAD« in einem Programm bewirkt nicht nur, wie im Direktmodus, das Laden eines Programms, sondern gleichzeitig einen Autostart. Falls das nachgeladene Programm größer ist als das ursprüngliche Programm, dann werden alle Variablen gelöscht. Sie sollten daher dafür sorgen, daß Ihr Hauptprogramm immer größer ist als das nachzuladende. Auch sollten Sie beachten, daß Strings in der Regel nicht mit übernommen werden. Wollen Sie Stringvariable dennoch ins nächste Programm übernehmen, dann müssen Sie beispielsweise im ersten Programm schreiben: A$="HALLO"+""

Dies ist deshalb nötig, weil nur Stringvariable, die mit irgendeiner Stringoperation (+, LEFT$,MID$, RIGHT$) verknüpft wurden, ihren Wert auch im nachgeladenen Programm behalten.

Um die manchmal unerwünschte Systemmeldung wie »Searching for« am Bildschirm nicht sichtbar werden zu lassen, gibt es eine sehr einfache Methode: Setzen Sie vor den LOAD-Befehl im Programm einfach die Schreibfarbe auf die Hintergrundfarbe. Sie verhindern damit zwar nicht die Ausgabe dieser Meldung, sie erscheint aber nicht sichtbar auf dem Bildschirm.

MSE-Probleme?

Es gelingt mir nicht, mit dem MSE eingegebene Programme nach dem Speichern und Wiederladen laufen zu lassen. Es ist überhaupt kein Programm mehr vorhanden. Nach LIST erscheint sofort »Ready«, ebenso nach RUN.

STEFAN GASS

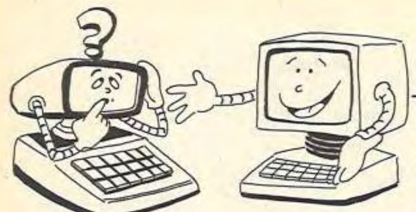
Der MSE dient zur einfachen und sicheren Eingabe von Programmen, die in Maschinensprache geschrieben sind. Maschinenprogramme lassen sich von Basic aus nicht LISTen und müssen mit dem SYS-Befehl statt mit RUN gestartet werden. Allerdings muß man dazu die Startadresse kennen. Diese Startadresse müssen Sie der Anleitung zum Programm entnehmen. Lesen Sie sich bitte zu jedem Maschinenprogramm die Anleitung sehr genau durch.

RUN = Syntax Error?

Läßt sich beim C 64 der Basic-Start verändern? Das Vorhandensein der Routine $A68E läßt mich dies zwar vermuten, es gelingt mir aber nicht ganz. Nach dem Heraufsetzen des Basic-Starts wird »RUN« mit »Syntax Error« beantwortet, ebenso »NEW«.

GERMANN PODEST

Der Syntax Error bei RUN ist schon ziemlich symptomatisch. Sie haben wahrscheinlich nicht bedacht, daß das erste Byte im Basic-Bereich immer ein Null-Byte sein muß. Vor dem SYS-Befehl müssen Sie also unbedingt einen POKE <Anfangsadresse>, 0-Befehl geben.



Diskette versehentlich formatiert?

Ich habe versehentlich eine Diskette mit vielen Programmen darauf formatiert. Wie kann ich diese Programme wieder zurückholen?

HINNERK BEHN

Wenn Sie die Diskette ohne Angabe einer ID formatiert haben (die Formatierung kann dann nur einige Sekundenbruchteile gedauert haben), dann ist nur das Directory gelöscht. Mit einem Diskettenmonitor (und unserem Floppy-Kurs) könnten Sie bei etwas Erfahrung in der Lage sein, das Inhaltsverzeichnis wieder zu restaurieren. Da in der Regel aber mit ID formatiert wird, sind Ihre Programme wohl verloren. Das 1541-Laufwerk beschreibt nämlich beim normalen Formatieren alle Sektoren der Diskette mit Null-Bytes. Die ursprünglich darin enthaltene Information geht dabei genauso verloren wie beim Überspielen einer normalen Musik-Kassette.

Zufallszahlen mit Wiederholung

Beim Arbeiten mit der »Zufallsformel« X=INT(RND(1)*N) + 1, durch die Zufallszahlen von 1 bis N erzeugt werden, habe ich festgestellt, daß die so erzeugten »Zufallszahlen« gar keine solchen sind. Zumindest wiederholen sie sich nach einiger Zeit in gleicher Folge. Nun meine Fragen:

Kann man überhaupt von Zufallszahlen reden, wenn sie sich nach einer gewissen Zeit in gleicher Folge wiederholen?

Kann man diese Tatsache umgehen, vielleicht durch Änderung der Formel?

Wie funktioniert eigentlich die Auswahl der Zufallszahlen?

MARCO HESSELBART

Entscheidenden Einfluß auf die erzeugten Zufallszahlen hat die Wahl des oft als unwichtig betrachteten Arguments der RND-Funktion. Generell sind drei Fälle zu unterscheiden:

Fall 1. Die Funktion wird mit einem positiven Argument aufgerufen, also beispielsweise mit RND(1). Anstelle der 1 kann eine beliebige andere Zahl oder Variable größer als Null stehen. In diesem Falle werden die Zufallszahlen nach einem bestimmten mathematischen Algorithmus, also nach einer bestimmten Rechenvorschrift, ermittelt. In diesem Falle kommen die Zufallszahlen (oder besser die Pseudo-Zufallszahlen), die nach dem Einschalten des Computers erzeugt werden, jedesmal in genau der gleichen Reihenfolge wieder vor, was sicherlich als nachteilig zu bewerten ist. Der Vorteil dabei ist jedoch die sehr gute Gleichverteilung der erzeugten Zahlen. Es werden weder Zahlen bevorzugt noch benachteiligt.

2. Die Funktion wird mit einem negativen Wert aufgerufen (Beispiel: RND(-1) oder RND(X), falls X<0). In diesem Fall ergibt jede einzelne negative Zahl einen speziellen Zufallswert, aber eben für jede Zahl stets wieder denselben. Probieren Sie ruhig zehnmal hintereinander RND(-1) aus, Sie erhalten stets die gleiche Zahl. Haben Sie aber einmal einen als zufällig geltenden Anfangswert X, dann können Sie mit der Folge »10 X=RND(-X) : PRINT X : GOTO 10« eine beliebige Kette von zufälligen Zahlen erzeugen. Jeder verschiedene Anfangswert für X liefert bei dieser Methode eine eigene, ganz spezielle Zufallszahlenreihe.

Als Startwert kommen alle möglichen Werte in Betracht, sinnvoll ist in vielen Fällen eine Initialisierung über die Zeitvariable TI. Der Startwert ist dann von der seit dem Einschalten vergangenen Zeit abhängig und kann in den meisten Fällen wohl als ausreichend zufällig angesehen werden.

Fall 3. Die Funktion wird mit dem Argument Null aufgerufen. In diesem Falle wendet der Basic-Interpreter eine weitere Methode an: Er nimmt die Zeiten verschiedener interner Zeitgeber, verknüpft sie durch eine mathematische Formel miteinander und bildet so ebenfalls eine Zufallszahl.

Diese Zahlen sind wirklich alle paar Mikrosekunden völlig anders und liefern keine vorhersehbare Folge von Zahlen. Allerdings ist bei dieser Methode die Gleichverteilung der Zahlen um einiges schlechter, jedenfalls wenn die Zufallszahlen in einer festen Programmschleife, also in einem festgelegten Zeittakt gebildet werden. In diesem Falle werden fast unweigerlich bestimmte Zahlenwerte häufiger vorkommen als andere.

Die effektivste Methode für möglichst zufällige, aber eben auch möglichst gleichverteilte Zahlen ist die zweite. Besonders bei Initialisierung mit TI; womöglich noch durch Multiplikation oder Division mit irgendwelchen Zeropage-Inhalten verknüpft, erreicht man die besten Ergebnisse.

Copyright-Problem

Darf ich eigentlich Programmteile, die im 64'er abgedruckt sind, für meine eigenen Programme benutzen und diese dann wieder einschicken?

RENE FREHNER

Da der Verlag Markt und Technik das Copyright an diesen Programmen besitzt, dürfen Sie auch einzelne Programmteile verwenden, wenn das schriftliche Einverständnis von Markt und Technik vorliegt, falls Sie Ihr Programm einem anderen Verlag anbieten.

Schicken Sie Ihr Listing aber an uns ein, dürfen Sie dieses Einverständnis natürlich immer voraussetzen.

Generell dürfen Sie einem Verlag Listings nur dann anbieten, wenn alle Rechte an dem Programm bei Ihnen liegen, das heißt in der Regel, wenn Sie es selbst geschrieben haben. Falls das Programm nur eine umgeschriebene Version eines anderen ist, oder falls einzelne Routinen aus anderen Quellen stammen, müssen Sie das unbedingt im Anschreiben erwähnen, sonst kann das unangenehme Folgen haben.

Falls Sie das Programm jedoch ausschließlich für Ihren privaten Gebrauch verwenden, dann brauchen Sie sich um solche Dinge keine Gedanken zu machen. Es entspricht aber einer guten Programmierer-Tradition und auch der allgemeinen Fairneß, bei aus anderen Programmen entnommenen Routinen die Original-Quelle und den Namen des Autoren der Routine anzugeben.

Zeichensatz verschwunden?

Aus welchem Speicherblock kann man die Daten eines geänderten Zeichensatzes PEEKen, wenn der Zeichensatz nach $E000 und das Video-RAM nach $CC00 verlegt worden ist? Ich habe den ganzen Speicher durchsucht, aber nichts gefunden.

JENS SCHLINGMANN

Ab Adresse $E000 sind die 8 KByte ROM des Betriebssystems dem RAM-Bereich überlagert. Während man mit POKE auch in diesem Bereich ins »unter« dem ROM liegende RAM schreiben kann, ergibt jedes PEEKen hier nur den Inhalt des Betriebssystem-ROMs. Abhilfe ist hier nur über Maschinensprache möglich, da zum Auslesen dieses Speicherbereichs das Betriebssystem und der Basic-Interpreter abgeschaltet werden muß.

Laden mit Bild?

Während der C64 ein Programm von Kassette lädt, zeigt er normalerweise kein Bild auf dem Monitor an. Nun habe ich aber an einigen Maschinensprache-Programmen gesehen, daß es doch möglich ist, ein Bild während des Ladens stehen zu lassen. Ist dieser Effekt auch in Basic zu erzielen oder ist dazu die Kenntnis von Maschinensprache notwendig?

DIETER KURBJUHN

Vom Basic aus ist das nicht möglich. Da der Videochip und der Prozessor abwechselnd auf denselben Datenbus zugreifen, kann es zu Zeitproblemen kommen, die das System zum Absturz bringen könnten. Um das zu verhindern, schaltet das Betriebssystem während der Dauer des Kassettenzugriffs den Videochip einfach aus, wodurch natürlich das Bild verschwindet.

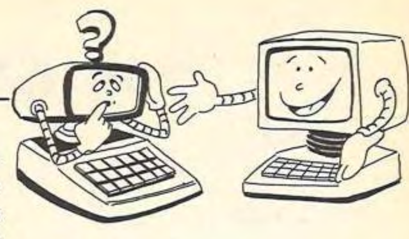
Um das zu verhindern, darf man nicht auf die vorhandenen Betriebssystemroutinen zugreifen – wie es das Basic automatisch macht, sondern muß sich eigene Laderoutinen schreiben.

Das allerdings ist nur in Maschinensprache möglich.

Wieviel Strom braucht der C64?

Welche Stromkosten verursacht ein einen Monat lang ununterbrochen laufender C64? Bitte in DM und als Vergleich (zum Beispiel »soviel wie eine 100-Watt-Glühbirne«).

Der C64 hat genau 15 Watt Leistungsaufnahme (steht jedenfalls auf dem Computer).



Die elektrische Arbeit in kWh (Kilowattstunden) bemißt sich aus Leistung mal Zeit. Wenn Sie das Floppy-Laufwerk auch noch mit einbeziehen (50 W), haben Sie einen Verbrauch von 46,8 kWh im Monat. Das einzige, was Sie noch tun müssen, ist, Ihr Elektrizitätswerk nach dem Preis einer kWh zu fragen. Rechnet man grob etwa 20 Pfennige pro kWh, dann kostet der Dauerbetrieb von Computer und Floppy noch keine 10 Mark im Monat. Allerdings ist darin noch nicht der Stromverbrauch des ja wohl in den meisten Fällen ebenfalls eingeschalteten Monitors oder Fernsehgerätes berücksichtigt.

Negative Bytes

Auf die Eingabe »PRINT FRE(0)« antwortet mein C64 stets mit einer negativen Zahl. Wie erhalte ich den tatsächlich freien Speicherplatz angezeigt?

KLAUS KRÖGER

Die FRE-Funktion liefert beim C64 stets eine Integer-Zahl (16 Bit) zurück. Der größte so korrekt darstellbare Wert ist 32768 ($=2^{15}$). Überschreitet der zur Verfügung stehende Speicherplatz diesen Wert, dann wird auch noch das Vorzeichenbit herangezogen, und es ergeben sich negative Zahlen. Das korrekte Ergebnis erhält man in diesen Fällen, indem man 65535 ($=2^{16}$) zum Ergebnis der FRE-Funktion addiert, also mit »? FRE(0) + 2↑16«.

Programme nachladen?

Wenn ein Basic-Programm ein anderes nachlädt, dann entsteht ein heilloses Durcheinander. Wie kann das kommen?

NORBERT BURGHART

Beim Nachladen von Programmen innerhalb eines anderen Programms werden die Zeiger, die auf das Programmende zeigen, nicht ordnungsgemäß nachgestellt. Falls das nachgeladene Programm länger ist als das erste Programm, dann kann es Probleme geben. Werden nämlich Variablen definiert, dann überschreiben sie das Programm. Es gibt nun zwei Möglichkeiten, dies zu verhindern:

1. Man lädt das Programm im Direktmodus. Dazu wird einfach eine Eingabe simuliert. Das geht folgendermaßen:

```
10 PRINT "CLR,3DOWN, 4SPACE" ; CHR$(34); "NAME" ; CHR$(34); " ,8"
20 PRINT "HOME"
30 POKE 631,131 : POKE 198,1
```

Zeile 30 simuliert die Eingabe der Tastenkombination SHIFT-RUN/STOP. Um zu vermeiden, daß man den Text auf dem Bildschirm lesen kann, muß nur die gleiche Farbe für Schrift und Hintergrund gewählt werden:

```
5 POKE 646, PEEK(53281)
```

2. Laden vom laufenden Programm aus mit Neusetzen der Zeiger. Die Zero-Page-Adressen 174 und 175 werden beim Laden eines Programms vom Betriebssystem benutzt und enthalten nach dem Laden die Endadresse des Programms. Wichtig für den Basic-Interpreter sind aber die Adressen 45,46. Sie zeigen ebenfalls auf das Programmende und auf den Start der Variablen. Am Anfang des nachgeladenen Programms sollte also stehen:

```
0 POKE 45, PEEK(174) : POKE 46, PEEK(175) : CLR
```

Durch CLR werden zwar leider alle Variablen gelöscht, es ist aber dennoch wichtig, denn es werden dadurch noch einige andere Zeiger korrigiert.

MICHAEL SIEPMANN

Probleme mit C128

Als neuer Besitzer eines C128 bin ich auf zwei Probleme gestoßen, bei denen ich vermute, daß mein C128 nicht ganz in Ordnung sein könnte:

(1) Die programmierte Umschaltung auf die DIN-Tastatur mittels »POKE 1,0« (siehe Handbuch, Kapitel 4.1, Seite 2) klappt bei mir nicht.

(2) Zeichne ich im hochauflösenden Grafikmodus zum Beispiel zwei Kreise in einer bestimmten Farbe und verbinde diese dann mit einer Linie in einer anderen Farbe, so werden diese verschiedenen Farben nicht voneinander getrennt, sondern es wird gleich das ganze Umfeld der Schnittstelle neu eingefärbt. Was kann ich dagegen tun?

OLIVER HOBERT

Zunächst eine gute Nachricht: Ihr C128 ist, zumindest was die beschriebenen Symptome angeht, völlig in Ordnung.

(1) Versuchen Sie die Umschaltung mit »POKE 0,PEEK (0) AND 64 : POKE 1,0«.

(2) Dieser Effekt hat eine verblüffend einfache Ursache: Im hochauflösenden Grafikmodus kann der C128 wohl einzelne Grafikpunkte setzen oder löschen, es ist jedoch nicht möglich, jedem Punkt eine individuelle Farbe zu geben. Die Farbgebung im Hochauflösungs-Modus entspricht haargenau derjenigen im Textmodus, das heißt, es kann jede Zeichenposition (25 x 40 Positionen auf dem Bildschirm) eine Farbe bekommen.

Mit anderen Worten: Die gewählte Farbe kann nicht punktweise, sondern nur pro 8 x 8-Punkte-Bereich (eben gerade eine Zeichenposition) frei gewählt werden. Innerhalb einer 8 x 8-Zeichen-Matrix haben immer alle 64 Grafikpunkte die gleiche Farbe.

Ihr spezielles Problem läßt sich nur durch Verwendung des Multicolor-Modus lösen. In diesem Vielfarbenmodus kann jeder einzelne Grafikpunkt eine von vier möglichen Farben haben. Allerdings ist die Auflösung in diesem Modus mit 160 x 200 Punkten nur noch halb so groß wie im hochauflösenden Grafikmodus.

Kabel-Geheimnis?

Wozu ist das nichtisolierte, aus dem Datasettenstecker herausragende Kabel da?

HEIKO FEDERHENN

Bei dem Kabel handelt es sich um eine Erdungsleitung, die bei den großen CBM-Computern zur Abschirmung dient. Der C64 ist durch eine über die gesamte Platine gelegten Metallfolie aber bereits ausreichend abgeschirmt, so daß Sie dieses Kabel ohne Gewissensbisse direkt am Stecker abschneiden können.

Schäden durch Floppy-Beschleuniger?

(1) Schaden Floppy-Beschleuniger der Mechanik des Laufwerkes 1541? Wenn ja, wieviel stärker ist die Belastung mit Speed Dos Plus und mit Prologic Dos?

(2) Muß die 1541-Floppy waagerecht aufgestellt werden, oder ist es auch möglich, sie senkrecht auf die Seite zu stellen?

MARIO VEITH

(1) Durch die sogenannten Floppy-Beschleuniger wird ja nicht etwa die Drehzahl der Floppy erhöht, sondern nur die Übertragungsgeschwindigkeit über den seriellen Bus. Mechanische Schäden sind also nicht zu befürchten.

(2) Die 1541-Floppy ist für waagerechten Betrieb konzipiert und sollte auch nur so betrieben werden.

Chromdioxid für Datasette?

Mein Computer-Händler sagt, ich solle keine Chromdioxid-Kassetten für meine Datasette benutzen. Ist da was Wahres dran?

MICHAEL SPLETT

Chromdioxid erfordert eine andere Vormagnetisierung als Eisenoxid. Auf lange Sicht gesehen kann die Verwendung von Chromdioxid-Kassetten bei nicht dafür vorgesehenen Recordern auch eine geringere Lebensdauer des Tonkopfes zur Folge haben.

Wegen der niedrigen Aufzeichnungsrate von 300 Baud ist die Verwendung von Chromdioxid aber auch beim besten Willen nicht notwendig.

Floppy-Laufwerk reinigen?

Welche Möglichkeit habe ich, meine Floppy 1541 zu reinigen, um Lese- und Schreibfehlern vorzubeugen, und wie bekomme ich das Laufwerk 30 Sekunden lang zum Laufen, wenn ich eine Reinigungsdiskette benutzen will?

JACK REIS

Sie können das Laufwerk für 30 Sekunden in Bewegung halten, wenn Sie es einfach fünfmal hintereinander initialisieren:

```
OPEN 1,8,15 : FOR I = 1 TO 5 : PRINT#1,"I" : NEXT :
CLOSE 1
```

Von der übermäßigen Benutzung von Reinigungsdisketten möchten wir aber unbedingt abraten, da diese den Schreib-/Lesekopf mit der Zeit so verschleißen, daß er unbrauchbar wird. Greifen Sie daher bitte nur dann zur Reinigungsdiskette, wenn sehr häufig Lese- oder Schreibfehler auftreten und Sie sicher sind, daß dies am verschmutzten Schreib-/Lesekopf liegt.



Tips und Tricks für Einsteiger

Es ist für einen Einsteiger anfangs immer schwer, aus seinem Computer die optimale Leistung herauszuholen. Die hier vorgestellten Routinen und Programme sollen Ihnen helfen, mit Ihrem Computer besser »auszukommen«.

Wir haben ein paar Leckerbissen für Sie herausgepickt: neben einem AUTO-Number, Listschutzmethoden und einem besseren INPUT-Befehl finden Sie unter anderem auch noch einen OLD-Befehl und einen Peripherietest.

Hilfe nach NEW oder RESET

Es kann schon mal vorkommen, daß man NEW eingibt und danach erschrocken feststellt: es wurde vergessen, das Programm zu speichern. Normalerweise sind diese Daten dann verloren. Mit Hilfe des Programms »Unnew« (Listing 1) kann jedoch das alte Programm wieder gerettet werden. Dazu müssen Sie jedoch das Hilfsprogramm schon vor dem NEW auf einer Diskette (oder Kassette) parat haben. Geben Sie bitte Listing 1 ein und legen eine formatierte Diskette (oder Kassette) in Ihr Laufwerk. Nach anschließendem Start mit RUN wird daraus ein Maschinenprogramm erzeugt und selbständig auf Ihren Datenträger gespeichert.

Ist Ihnen nun ein Mißgeschick passiert (NEW eingegeben?), so legen Sie die Diskette mit dem Unnew-Programm ein, laden dieses mit

```
LOAD"UNNEW",8,1
```

und starten es durch

```
SYS 525
```

Ihr Basic-Programm wird damit gerettet, selbst wenn Sie einen Reset ausgelöst haben sollten.

(Daniel Kossmann/dm)

Directory ohne Programmverlust

Gelegentlich möchte man das Directory einer Diskette ansehen, ohne ein eventuell im Speicher befindliches Programm zu zerstören. Befindet sich das DOS 5.1 (Erleichtert den Umgang mit der Floppy – zu finden auf Ihrer Test-/Demodiskette und im Artikel in diesem Heft) gerade nicht im Speicher, so behilft man sich meist mit der zeitaufwendigen Zwischenspeicherung des Programms auf Diskette. Es geht jedoch auch anders. Geben Sie einfach den folgenden Befehl im Direktmodus ein:

```
POKE44,PEEK(46)+1
```

Damit wird der Basic-Anfang auf einen freien Speicherbereich gesetzt. Sie können jetzt wie gewohnt mit

```
LOAD"$",8
```

das Directory laden und anschließend LISTen. Mit

```
POKE44,8
```

gelangen Sie dann wieder zurück in das eigentliche Programm.

Eine zweite, komfortablere Methode ist, das folgende kleine Programm (Listing 2) in sein eigenes Programm einzubauen und je nach Bedarf dieses dann auszuführen (zum Beispiel als Unterroutine, die mittels GOSUB aufgerufen werden kann – dabei muß das END in Zeile 20 durch ein RETURN ersetzt werden).

(Heinzpeter Oelkers/dm)

```
10 OPEN 1,8,0,"$":GET#1,A$,A$                       <124>
20 GET#1,A$,A$:IF ST=64 THEN CLOSE 1:END             <007>
30 GET#1,A$,B$:PRINT ASC(A$+CHR$(0))+256*A
   SC(B$+CHR$(0));                                   <078>
40 GET#1,A$:PRINT A$;:IF A$<>""THEN 40               <109>
50 PRINT:GOTO 20                                     <083>
```

Listing 2. Das Directory ausgeben, ohne ein Programm zu zerstören

```
100 FOR I=525 TO 578                   <086>
110 READ A:POKE I,A:NEXT I             <156>
120 POKE 43,525 AND 255:POKE 44,2      <135>
130 POKE 45,578 AND 255:POKE 46,2      <056>
140 CLR:SAVE"UNNEW",8,1                <138>
150 REM FUER CASSETTENBETRIEB:         <117>
160 REM CLR:SAVE"UNNEW",1,1            <095>
170 DATA 160,3,200,177,43,208          <103>
180 DATA 251,200,200,152,160,0         <203>
190 DATA 145,43,165,44,200,145         <194>
200 DATA 43,133,60,160,0,132           <195>
210 DATA 59,162,0,200,208,2,230        <206>
220 DATA 60,177,59,208,245,232         <092>
230 DATA 224,3,208,242,200,208         <018>
240 DATA 2,230,60,132,45,164,60        <101>
250 DATA 132,46,96,255                 <065>
```

Listing 1. Ein Programm, das Ihnen bei NEW oder RESET hilft

Mehr Diskettenkapazität

So können Sie die Kapazität Ihrer Diskette verdoppeln und Ihre Kosten halbieren: Sie müssen nur auf der anderen Seite noch mal genau die gleiche Schreibschutzöffnung einschneiden. Das geht am besten mit einem Papierlocher (siehe Bild 1).

(dm)

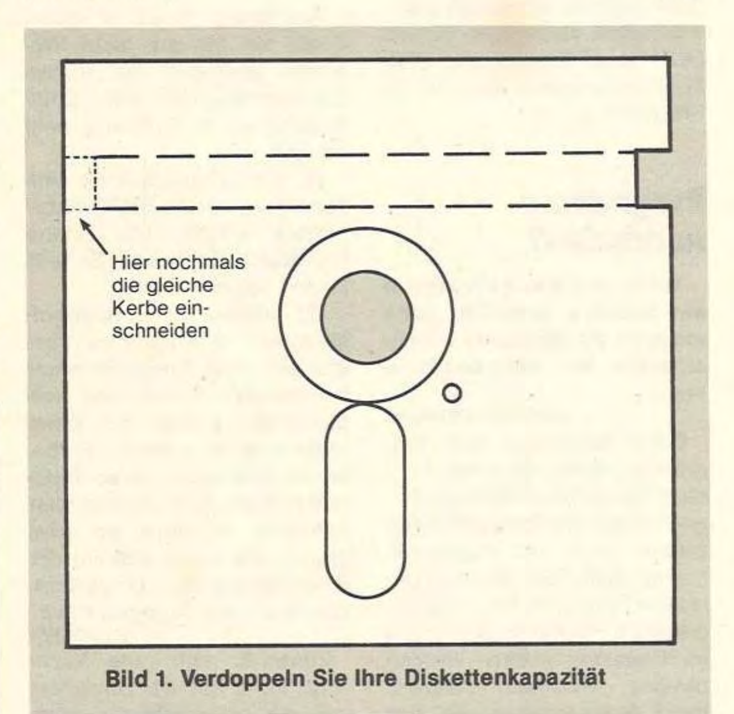


Bild 1. Verdoppeln Sie Ihre Diskettenkapazität

Listschutz für Basic-Programme

Ein Listschutz dient dazu, sein Programm und seinen Programmierstil vor fremden Augen zu schützen.

Möchte man ein Basic-Programm mit einem einfachen Listschutz versehen, so verfährt man folgendermaßen:
1. Man ergänzt die Zeile, ab der der Listschutz wirksam werden soll, mit »:REM" "«.
2. Man fährt mit dem Cursor auf das zweite Anführungszeichen und drückt fünfmal die Tastenkombination SHIFT-INST.
3. Nun wird ebenfalls fünfmal die Taste DEL gedrückt, so daß zwischen den Anführungszeichen fünf reverse T stehen.
4. Zuletzt bewegt man den Cursor hinter das zweite Anführungszeichen und drückt die Tastenkombination SHIFT und L. Anschließend RETURN nicht vergessen.

Versucht man nun, das Programm zu listen, gelangt der Computer nur bis zu der Zeile, in der der Listschutz steht und bricht den Vorgang dann mit der Meldung SYNTAX ERROR ab.

Dieser Listschutz ist allerdings relativ leicht wieder aufzuheben.

(Thomas Lopatic/dm)

Ein geänderter Zeichensatz

Mit Hilfe dieses Programmes kann man auf ganz einfache Art einen neuen Zeichensatz (etwas dünner als der gewohnte) erstellen. Nach einer kleinen Wartezeit, in der dieser aufgebaut wird, können Sie über ein neues Schriftbild verfügen.

```
10 R=56334:POKE 53272,24:POKE R,0
20 POKE 1,51:FOR I=6 5 TO R/5
30 POKE I,PEEK(I+45056) AND 60
40 NEXT:POKE 1,7:POKE R,1
```

(dm)

Negative Bytes?

Die FRE-Funktion liefert beim C64 gelegentlich negative Werte (negative Bytes?) Dies ist jedoch nur ein kleiner Fehler im Betriebssystem des Computers. Abhilfe schafft hierbei, zu der negativen Zahl den Wert 65535 zu addieren, also PRINT FRE(0)+65535. (dm)

Sequentielle Datei retten

Wenn auch nicht oft, so kann es doch gelegentlich einmal vorkommen, daß beim Schreiben einer sequentiellen Datei auf Diskette etwas passiert. Sei es, daß das Programm abstürzt oder durch einen Programmier- oder Bedienungsfehler die Datei nicht richtig geschlossen wurde. Durch geschickte Ausnutzung des COPY-Befehles läßt sich die Datei aber noch retten. Man muß nur im Direktmodus folgendes eingeben:

```
D$="NAME"  (Name der verlorenen Datei)
OPEN1,8,15
PRINT#1,"C:X="+D$
PRINT#1,"S:"+D$
PRINT#1,"R:"+D$+"=X"
CLOSE1
```

(Erwin Fröhlich/dm)

Kassettendirectory

Eine einfache, wenn auch sehr langwierige Methode, sich ein Kassettendirectory zu erstellen, ist folgende (ein Drucker wird vorausgesetzt): Geben Sie diese Direktbefehle über die Tastatur ein:

```
OPEN1,4:CMD1:LOAD"QAY"
```

oder einen anderen, wirren Namen, der bestimmt nicht auf der Kassette zu finden ist. Nach dieser Eingabe muß die Space-Taste festgeklemmt (etwa mit einem Stück Tesafilm) und der Recorder gestartet werden. Dieser läßt die Kassette durchlaufen und gibt jedesmal, wenn er etwas findet, ein FOUND PROGRAMMNAME auf dem Drucker aus. Dieses Verfahren funktioniert auch mit Beschleunigern wie »Turbo Tape« oder »Super Tape«. (dm)

Restore für Unterprogramme

Wenn beide Programmteile, Hauptprogramm und Unterprogramm, DATAs enthalten, muß sichergestellt werden, daß auch die richtigen Werte gelesen werden. Wenn man nicht aufpaßt, kann es passieren, daß das Unterprogramm DATAs aus dem Hauptprogramm liest. Wie läßt sich das verhindern? Es gibt eine umständliche Methode: Man kann eine kleine Basic-Erweiterung einbauen, den RESTORE X-Befehl. Es geht aber auch einfacher. Die Zeropage, das sind die ersten 256 Bytes des Speichers, hilft uns bei der Lösung des Problems. Genauer gesagt, die Adressen 65/66 und 122/123. Schlagen wir im Commodore-Handbuch nach, dann steht dort:

65 bis 66 Adresse des aktuellen DATA-Elementes
122 bis 123 Basic-Zeiger innerhalb der Subroutine

Mit diesen Informationen läßt sich schon etwas anfangen. Wenn das Unterprogramm angesprungen wird, dann sollte der Zeiger in Speicherstelle 122/123 auf die Adresse des Unterprogramms im Speicher stehen. POKEt man diese Werte in die Zeilen 65/66 mit

```
POKE 65,PEEK(122)
POKE 66,PEEK(123)
```

so wird beim nächsten READ der Wert gelesen, der hinter dieser Basic-Zeile mit den POKEs steht, also das erste DATA-Element innerhalb des Unterprogrammes. Nach dem Rücksprung aus dem Unterprogramm muß der Zeiger eventuell auch im Hauptprogramm wieder gestellt werden.

In dem kurzen Demo-Listing (Listing 3) werden drei Unterprogramme in zufälliger Reihenfolge aufgerufen.

(Stephan Pätzold/dm)

```
1 REM ***********************                <058>
2 REM *          DEMO         *              <161>
3 REM * SUBROUTINE/RESTORE *                 <075>
4 REM ***********************                <061>
5 PRINT"{CLR,6SPACE}TASTE DRUECKEN!"         <161>
6 PRINT:PRINT                                <214>
10 X=INT(RND(TI)*3)+1                        <235>
20 ON X GOSUB 1000,2000,3000                 <012>
25 POKE 65,PEEK(122):POKE 66,PEEK(123)       <084>
30 READ A$:PRINT A$                          <249>
50 DATA"HAUPTPROGRAMM"                       <118>
100 GOTO 10                                  <022>
999 :                                        <213>
1000 REM *** SUBROUTINE 1 ***                <208>
1005 :                                       <219>
1010 POKE 65,PEEK(122):POKE 66,PEEK(123)     <051>
1020 FOR I=1 TO 4:READ A:PRINT A;:NEXT       <051>
1030 READ A$:PRINT A$;                       <095>
1040 POKE 198,0:WAIT 198,1                   <252>
1050 DATA 1,11,111,1111,"UP 1 "              <118>
1060 RETURN                                  <102>
1070 :                                       <030>
2000 REM *** SUBROUTINE 2 ***                <193>
2005 :                                       <203>
2010 POKE 65,PEEK(122):POKE 66,PEEK(123)     <035>
2020 FOR I=1 TO 4:READ A:PRINT A;:NEXT       <035>
2030 READ A$:PRINT A$;                       <077>
2040 POKE 198,0:WAIT 198,1                   <234>
2050 DATA 2,22,222,2222,"UP 2 "              <214>
2060 RETURN                                  <086>
2070 :                                       <014>
3000 REM *** SUBROUTINE 3 ***                <178>
3005 :                                       <187>
3010 POKE 65,PEEK(122):POKE 66,PEEK(123)     <019>
3020 FOR I=1 TO 4:READ A:PRINT A;:NEXT       <019>
3030 READ A$:PRINT A$;                       <061>
3040 POKE 198,0:WAIT 198,1                   <218>
3050 DATA 3,33,333,3333,"UP 3 "              <051>
3060 RETURN                                  <068>
3070 :                                       <252>
```

Listing 3. Ein Demo-Programm zur Erklärung des »RESTORE X«

Fehler im Floppy-DOS

Wenn Sie schon einmal versucht haben, ein Programm mit der REPLACE-Funktion

```
(SAVE"@:NAME",8)
```

zu speichern und das Programm dann zerstört war, so liegt es an einem Fehler im Floppy-Betriebssystem, der allerdings sehr selten auftritt. Abhilfe schafft hier nur die Methode, das alte Programm erst zu löschen und dann das Neue zu speichern. (dm)

Neues INPUT

Haben Sie schon einmal versucht, professionelle Programme zu schreiben? Dann hat Ihnen mit Sicherheit die INPUT-Routine des C64-Betriebssystems zu schaffen gemacht, die sämtliche Zeichen zuläßt, auch solche, die zu völligem Unsinn oder Fehlermeldungen führen. Dieser Mißstand wird in Basic-Programmen meist mit einer GET-Schleife umgangen. Bei dieser Methode wird ständig ein Zeichen von der Tastatur geholt und auf Zulässigkeit überprüft. Die einzelnen Zeichen werden in einem String addiert. Wenn Sie mit INS oder DEL Korrekturen ausführen wollen, können Sie sich vorstellen, wie kompliziert und zeitraubend eine solche Routine werden kann.

Eine schnelle Lösung des Problems schafft dieses Maschinenprogramm (Listing 4), das einen neuen INPUT-Befehl generiert. Die Syntax des Befehls lautet:

INPUT > Eingabelänge, Spalte, Zeile, Variable

Wie Sie schon an der Syntax sehen, ein recht komfortabler INPUT-Befehl. Mit der Eingabelänge geben Sie an, wie lang das Eingabefeld höchstens sein darf. Durch Angabe von Zeile und Spalte können Sie das Eingabefeld direkt an jeder beliebigen Bildschirmstelle positionieren. Als Variable müssen Sie eine Stringvariable einsetzen, zum Beispiel A$. Sollen ausschließlich Zahlen eingegeben werden, braucht der String nur noch auf Buchstaben hin überprüft werden. Beispielsweise mit folgender Routine:

```
10 INPUT>4,1,1,A$
20 IF STR$(VAL(A$))=" "+A$ THEN PRINT"ES IST EINE ZAHL":END
30 PRINT"KEINE ZAHL":GOTO10
```

Tips zur Eingabe

Das Programm (Listing 4) müssen Sie mit dem MSE eingeben und speichern. Den MSE finden Sie auf Seite 136 in dieser Ausgabe. Laden Sie dann die neue Routine mit

LOAD"E-ROUTINE 64",8,1 (für die Floppy)

oder

LOAD"E-ROUTINE 64",1,1 (für Datasette)

und geben Sie

NEW

und

SYS 49152

zur Initialisierung ein. Danach können Sie Ihre Basic-Programme eintippen und mit dem neuen INPUT-Befehl verwenden. Die Routine können Sie natürlich auch mit dem Basic-Programm nachladen und mit dem neuen INPUT-Befehl verwenden.

Zusammenfassung

- Die Routine wird mit SYS 49152 aktiviert.
- Das Programm benötigt den Speicherbereich von Adresse 49152 bis 49617
- Die Syntax lautet: INPUT > Eingabelänge, Zeile, Spalte, Stringvariable
- Die CLR-; HOME-; RUN-; STOP- und SHIFT-Taste ist bei der Eingabe gesperrt.
- Das Eingabefeld kann mit den Cursortasten nicht verlassen werden.
- INST/DEL wirkt nur innerhalb des Eingabefeldes.
- Leerzeichen am Ende der Eingabe werden nicht mit übernommen.
- Benötigen Sie Leerzeichen am Schluß, geben Sie einfach POKE 49298,131 ein. POKE 49298,94 schaltet wieder in den Normalzustand.

Ein Nachteil soll nicht verschwiegen werden: Fragetext kann nicht wie beim serienmäßigen INPUT-Befehl ausgegeben werden. Der Text muß mit PRINT gedruckt werden. Das bedeutet, Sie müssen erst den Text mittels PRINT ausgeben und dann quasi den INPUT »drüberlegen«.

Eine komfortablere Routine finden Sie neben anderen Befehlen in der Basic-Erweiterung »Datawork-Basic«, erschienen in der 64'er, Ausgabe 1/86.

(Thomas Graf/dm)

```
programm : e-routine 64 c000 c1f4
----------------------------------
c000 : a9 0b a0 c0 8d 08 03 8c  ad
c008 : 09 03 60 20 73 00 c9 85  18
c010 : f0 06 20 79 00 4c e7 a7  8c
c018 : 20 73 00 c9 b1 f0 06 20  26
c020 : bf ab 4c ae a7 a9 00 8d  81
c028 : 6e c1 20 9b b7 8e 6f c1  23
c030 : 20 fd ae 20 9e b7 e0 28  7a
c038 : b0 0c 86 fd 20 fd ae 20  3d
c040 : 9e b7 e0 19 90 03 4c 48  f8
c048 : b2 8e 70 c1 a4 fd 20 0c  68
c050 : e5 20 24 ea a5 d2 85 fe  b1
c058 : 98 18 65 d1 90 02 e6 fe  43
c060 : 85 fd a5 f4 85 fc 98 18  bf
c068 : 65 f3 90 02 e6 fc 85 fb  90
c070 : a5 c6 85 cc 8d 92 02 f0  cb
c078 : f7 78 a5 cf f0 0c a5 ce  b2
c080 : ae 87 02 a0 00 84 cf 20  2a
c088 : 13 ea 20 b4 e5 c9 0d d0  31
c090 : 03 4c 71 c1 c9 14 f0 46  db
c098 : c9 9d f0 42 c9 1d f0 29  50
c0a0 : c9 94 f0 61 c9 91 d0 03  8e
c0a8 : 4c 41 c1 c9 11 d0 03 4c  7b
c0b0 : 54 c1 ae 18 d0 e0 15 f0  de
c0b8 : 08 c9 db b0 b3 c9 c1 b0  a4
c0c0 : 08 c9 60 b0 ab c9 20 90  86
c0c8 : a7 c9 22 f0 a3 ae 6e c1  e7
c0d0 : ec 6f c1 f0 9b ee 6e c1  71
c0d8 : 20 16 e7 4c 70 c0 ac 6e  23
c0e0 : c1 f0 8d ce 6e c1 c9 9d  ae
c0e8 : f0 ee b1 fb aa b1 fd 88  7c
c0f0 : 91 fd 8a 91 fb c8 c8 f0  60
c0f8 : 07 cc 6f c1 90 ec f0 ea  83
c100 : a9 9d 4c d8 c0 ac 6f c1  59
c108 : ce 6f c1 88 c0 ff f0 2b  35
c110 : cc 6e c1 90 26 b1 fd c9  11
c118 : 20 f0 f0 cc 6f c1 f0 1b  85
c120 : b1 fb aa b1 fd c8 91 fd  18
c128 : 8a 91 fb 88 88 c0 ff f0  fb
c130 : 05 cc 6e c1 b0 ea c8 a9  48
c138 : 20 91 fd ee 6f c1 4c 70  95
c140 : c0 aa 38 ad 6e c1 e9 28  06
c148 : 90 07 8d 6e c1 8a 4c d8  e0
c150 : c0 4c 70 c0 aa 18 ad 6e  69
c158 : c1 69 28 b0 07 cd 6f c1  0e
c160 : f0 05 90 03 4c 70 c0 8d  be
c168 : 6e c1 8a 4c d8 c0 ea ea  f8
c170 : ea 4c 96 c1 88 c0 ff d0  8e
c178 : 13 20 73 00 c9 00 f0 07  e7
c180 : c9 3a f0 03 4c 79 c1 a0  dc
c188 : 00 4c eb c1 b1 fd c9 20  54
c190 : f0 e2 c8 8c 6f c1 20 fd  36
c198 : ae 20 8b b0 48 98 48 20  fa
c1a0 : a3 b6 68 85 65 68 85 64  e2
c1a8 : ad 6f c1 20 75 b4 84 fb  88
c1b0 : a0 00 91 64 c8 8a 91 64  31
c1b8 : c8 a5 fb 91 64 a0 01 b1  37
c1c0 : 64 48 c8 b1 64 85 65 68  89
c1c8 : 85 64 ac 6f c1 88 b1 fd  bb
c1d0 : ae 18 d0 e0 15 f0 08 c9  67
c1d8 : 41 90 04 09 80 d0 06 c9  be
c1e0 : 20 b0 02 09 40 91 64 c0  9e
c1e8 : 00 d0 e2 ae 70 c1 20 0c  8c
c1f0 : e5 4c ae a7 00 00 00 00  9c
```

**Listing 4.
Ein komfortabler INPUT-Befehl**

Hinweis zum Abtippen
Dieses Listing muß mit dem MSE eingegeben werden. Den MSE finden Sie in dieser Ausgabe auf Seite 136.

```
 PROGRAMM : EYSSELE          C900 CBD7
-----------------------------------------
C900 : A9 5A A0 C9 8D 1A 03 8C  06
C908 : 1B 03 A9 91 A0 C9 8D 1C  08
C910 : 03 8C 1D 03 A9 AD A0 C9  1F
C918 : 8D 1E 03 8C 1F 03 A9 C8  49
C920 : A0 C9 8D 20 03 8C 21 03  2B
C928 : A9 E3 A0 C9 8D 26 03 8C  53
C930 : 27 03 A9 FF 8D 03 DD AD  07
C938 : 02 DD 09 04 8D 02 DD 60  0D
C940 : 48 A9 10 2C 0D DD F0 FB  62
C948 : 68 8D 01 DD AD 00 DD 09  D7
C950 : 04 8D 00 DD 29 FB 8D 00  7F
C958 : DD 60 A6 B8 F0 05 20 0F  FC
C960 : F3 D0 03 4C FE F6 A6 98  79
C968 : E0 0A 90 03 4C FB F6 E6  20
C970 : 98 A5 B8 9D 59 02 A5 B9  6C
C978 : 09 60 9D 6D 02 A5 BA 9D  3A
C980 : 63 02 C9 04 F0 04 C9 10  4E
C988 : 90 02 18 60 C9 00 4C 77  E8
C990 : F3 20 14 F3 F0 02 18 60  57
C998 : 20 1F F3 8A 48 A5 BA C9  C6
C9A0 : 10 B0 07 C9 04 F0 03 4C  70
C9A8 : 9D F2 4C F1 F2 20 0F F3  64
C9B0 : F0 03 4C 01 F7 20 1F F3  3A
C9B8 : A5 BA C9 04 F0 04 C9 10  24
C9C0 : 90 03 4C 0A F7 4C 19 F2  52
C9C8 : 20 0F F3 F0 03 4C 01 F7  11
C9D0 : 20 1F F3 A5 BA C9 04 F0  1D
C9D8 : 04 C9 10 90 03 4C 75 F2  25
C9E0 : 4C 5B F2 48 85 9E A5 9A  B9
C9E8 : C9 10 B0 07 C9 04 F0 03  4D
C9F0 : 4C CD F1 98 48 8A 48 A5  F7
C9F8 : 9E A4 9A C0 10 D0 06 20  87
CA00 : 40 C9 18 90 1F C0 11 D0  1B
CA08 : 06 20 5C CA 18 90 15 C0  6B
CA10 : 04 F0 04 C0 12 D0 06 20  A5
CA18 : 2B CA 18 90 07 C0 13 D0  25
CA20 : 03 20 44 CA 68 AA 68 A8  6D
CA28 : 68 18 60 C9 41 90 12 C9  62
CA30 : 5F B0 04 09 20 D0 0A C9  4E
CA38 : C1 90 06 C9 DE B0 02 29  CA
CA40 : 7F 4C 40 C9 C9 FF F0 18  BF
CA48 : C9 60 B0 03 4C 40 C9 E9  90
CA50 : 40 10 02 E9 40 A0 D0 84  AB
CA58 : 06 4C C4 CA C9 FF D0 06  FB
CA60 : A2 5E A0 D0 D0 5B 48 A4  C6
CA68 : B9 C0 FF D0 02 E6 B9 29  2C
CA70 : 7F C9 20 90 2C A8 A5 B9  00
CA78 : 29 01 F0 10 68 C9 A0 90  D8
CA80 : 04 C9 C0 90 03 4C 2B CA  80
CA88 : E9 40 D0 0E 68 C9 60 B0  3F
CA90 : 03 4C 40 C9 E9 40 10 02  E7
CA98 : E9 40 A0 D0 84 06 4C C4  17
CAA0 : CA 68 24 0F 30 03 4C 40  56
CAA8 : C9 18 69 40 30 02 69 40  19
CAB0 : AA A5 B9 29 02 D0 71 A0  6E
CAB8 : D0 A5 B9 29 01 F0 02 A0  CF
CAC0 : D8 84 06 8A A8 A9 00 A2  CA
CAC8 : 07 9D C0 02 CA 10 FA 98  58
CAD0 : 4A 4A 4A 4A 4A 18 65 06  22
CAD8 : 85 06 98 0A 0A 0A 85 05  D9
CAE0 : A9 01 85 03 78 A5 01 29  D7
CAE8 : FB 85 01 A0 07 B1 05 85  17
CAF0 : 02 A2 07 06 02 90 08 BD  06
CAF8 : C0 02 05 03 9D C0 02 CA  D9
CB00 : 10 F1 06 03 88 10 E6 A5  DB
CB08 : 01 09 04 85 01 58 A2 00  9D
CB10 : BD D2 CB 20 40 C9 E8 E0  E5
CB18 : 05 D0 F5 A2 07 BD C0 02  BC
CB20 : 20 40 C9 CA 10 F7 18 60  0E
CB28 : E0 D5 90 01 CA 8A 29 7F  DC
CB30 : C9 50 90 02 E9 03 C9 45  EE
CB38 : 90 02 E9 03 C9 41 90 02  91
CB40 : E9 21 C9 1C 90 02 E9 08  81
CB48 : C9 11 90 02 E9 0B 38 E9  AA
CB50 : 04 AA A9 3C 20 40 C9 A0  08
CB58 : FF CA F0 08 C8 B9 7A CB  D5
CB60 : 10 FA 30 F5 C8 B9 7A CB  94
CB68 : 30 06 20 40 C9 18 90 F4  35
CB70 : 29 7F 20 40 C9 A9 3E 4C  E4
CB78 : 40 C9 57 48 D4 43 52 C4  B6
CB80 : 52 4F CE 48 4F CD 52 45  6E
CB88 : C4 43 52 D2 47 52 CE 42  A3
CB90 : 4C D5 4F 52 C7 46 B1 46  E7
CB98 : B3 46 B5 46 B7 46 B2 46  A9
CBA0 : B4 46 B6 46 B8 42 4C CB  54
CBA8 : 43 52 D5 52 4F C6 43 4C  A5
CBB0 : D2 42 52 CE 48 52 D4 47  0B
CBB8 : 52 B1 47 52 B2 48 47 CE  27
CBC0 : 48 42 CC 47 52 B3 50 55  F4
CBC8 : D2 43 52 CC 59 45 CC 43  E3
CBD0 : 59 CE 1B 2A 04 08 00 FF  1D
```

Listing 5. Ein preiswertes und sehr leistungsfähiges Software-Interface. Bitte mit dem MSE eingeben.

Ein Centronics-Interface

Ein Interface wird benötigt, um die Datenübertragung von einem C64 auf Nicht-Commodore-Drucker (zum Beispiel Epson FX-80/85) zu ermöglichen. Dazu muß das Interface die vom Computer gesendeten Zeichen und Befehle so wandeln, daß der andere Drucker sie versteht und ausführt. Dies kann auf teuere Weise mittels eines Hardware-Interfaces oder billiger über eine Schnittstelle in Form eines Programmes erfolgen (Hardware-Interface bedeutet: In den Drucker muß ein meist teures Modul eingebaut werden, das diese Aufgabe übernimmt – wie etwa das Görlitz-, Wiesemann- oder HDS-Interface).

Die hier vorgestellte Software-Schnittstelle (unter Software versteht man Programme) übertrifft mit ihren Leistungsmerkmalen viele käufliche Hardware-Schnittstellen. Nicht aber im Preis: Nur etwa 20 Mark kostet diese »Selbstbau«-Centronics-Schnittstelle

Es handelt sich hierbei um ein Maschinenspracheprogramm mit 726 Byte Länge, das den Adreßbereich $C900 bis $CBD1 (dezimal: 51456 bis 52182) belegt (Listing 5). Ein Vorteil des Programmes besteht in der Verträglichkeit mit DOS 5.1. Das bedeutet: Es können beide Programme auf einmal im Speicher des Computers liegen. Die Befehle beider Routinen können eingegeben werden, ohne daß man mit Fehlern rechnen muß.

Das Programm ist ohne Einschränkung für alle Epson-kompatiblen Drucker anwendbar, die über einen Bitmustermodus verfügen. Dieser Modus wird für die Ausgabe der Commodore-eigenen Grafikzeichen benötigt. Das Programm besteht aus mehreren Programmteilen, von denen die meisten Erweiterungen bestehender Ein-/Ausgaberoutinen des Betriebssystems sind und bei der Initialisierung in diese eingebunden werden. Dadurch können schon bestehende Basic-Befehle wie OPEN und PRINT# zum Drucken benutzt und Programme müssen nicht umgeschrieben werden. Zur Ansteuerung verschiedener Druckmodi wurden jedoch zusätzliche Gerätenummern definiert, deren Bedeutung in Tabelle 1 erklärt ist.

Gerätenummer 16 realisiert einen sogenannten Direktmodus, mit dem die internen Commodore-Zeichencodes ohne Wandlung an den Drucker gelangen. Der Direktmodus ist zur Ausgabe von Steuerzeichen oder bei der Verwendung des

GERÄTENUMMER 16	=	DIREKTMODUS
GERÄTENUMMER 18,4	=	TEXTMODUS
GERÄTENUMMER 19	=	GRAFIKMODUS
GERÄTENUMMER 17	=	LISTMODUS

SEK.-ADR.	MODUS	STEUERZEICHEN
0	NORMAL	NORMAL
1	KLEIN	NORMAL
2	NORMAL	ERKLÄRT
3	KLEIN	ERKLÄRT

Tabelle 1. Zusätzliche Gerätenummern zur Ansteuerung verschiedener Druckmodi

USER PORT - CENTRONICS		
A	GND	16
B	FLAG - BUSY	11
C	D0	2
D	D1	3
E	D2	4
F	D3	5
H	D4	6
J	D5	7
K	D6	8
L	D7	9
M	PA2 - STROBE	1

Tabelle 2. Verbindungsplan von User-Port auf Centronics

Druckers als Plotter zur Einzelnadelsteuerung beziehungsweise zur Ausgabe von Bitmustern geeignet. Der Textmodus (Groß- und Kleinschreibung) ist unter Gerätenummer 18 und, weil er wohl am häufigsten bei bereits bestehenden Programmen benutzt wird, unter Gerätenummer 4 ansprechbar. Die Gerätenummer 19 realisiert den Großschrift-/Grafikmodus, wie er beim C64 gleich nach dem Einschalten voreingestellt ist. Der wichtige Modus zum Listen von Programmen wurde mit Gerätenummer 17 realisiert. Es läßt sich über die Sekundäradresse noch zwischen vier Fällen unterscheiden. Einmal kann gewählt werden, ob das Listing, wie vom Bildschirm her gewohnt, mit Großbuchstaben und Grafikzeichen oder im Textmodus mit großen und kleinen Buchstaben gedruckt wird. Zum anderen kann man wählen, ob die Steuerzeichen wie bei Bildschirmausgabe als inverse Zeichen oder durch



Abkürzungen wie <CLR> (Bildschirm löschen) im Klartext gedruckt werden. Durch all diese Möglichkeiten kann der Anwender zum einen auf seinem Drucker Ausgaben erzeugen, wie man sie von Commodore-Druckern her gewohnt ist, zum anderen auch alle Möglichkeiten seines Druckers voll nutzen.

Das Programm (Listing 5) müssen Sie mit dem MSE eingeben und auf Diskette oder Kassette speichern. Vor dem Start sollten Sie das Verbindungskabel Userport-Centronics zusammenlöten. Dazu ist in Tabelle 2 ein Verbindungsplan angegeben. Das Kabel sollte für eine störungsfreie Funktion nicht länger als einen Meter sein und aus abgeschirmtem, mehradrigem Steuerkabel bestehen, das man in (fast) jedem Elektronik-Bastelgeschäft findet. Dort sind in der Regel ebenfalls der Centronics-Stecker und der Stecker für den Userport erhältlich.

Initialisiert wird die ganze Treiberroutine mit

```
SYS 51456
```

Hierbei wird die Routine in das Betriebssystem eingebunden. Jedoch Vorsicht: Nach einem Abbruch des Programms, zum Beispiel durch die Betätigung der Tasten RUN/STOP und RESTORE ausgelöst, muß die Routine erneut initialisiert werden.

Die einzelnen Druckmodi spricht man mit den üblichen Basic-Befehlen an. Geöffnet wird der Ausgabekanal mit:

OPEN log.Dateinummer, Geräteadr. [,Sekundäradr.]

Die eckigen Klammern kennzeichnen optionale Angaben. Nach Öffnen des Kanals zum Drucker kann dann mit PRINT # log.Dateinummer der Text ausgegeben werden. Ein Programm-Listing wird zum Beispiel erzeugt mit den Befehlen

```
OPEN 17,17,0:CMD17:LIST
PRINT#17:CLOSE17
```

Der PRINT-Befehl vor dem CLOSE ist notwendig, damit der CMD-Modus aufgehoben wird.

Umstellen auf beliebige Drucker mit Centronics-Schnittstelle

Das Programm wurde für einen Epson-Drucker geschrieben. Unverändert ist es für jeden anderen Drucker mit Centronics-Schnittstelle verwendbar, sofern auf die Ausgabe von Commodore-eigenen Grafikzeichen verzichtet wird (dazu gehören auch die reversen Zeichen).

(H. Eyssele/dm)

Automatische Zeilennummernvorgabe

Das lästige Eingeben der Zeilennummern beim Programmieren kann Ihnen dieses kleine Programm abnehmen. Die Syntax des AUTO-Befehls lautet:

```
←A Anfangszeilennummer, Schrittweite
```

Nach Eingabe dieses Befehls wird die Zeilennummer vorgegeben und nach RETURN um »Schrittweite« erhöht. Um aus dem AUTO-Modus wieder herauszukommen, muß man nach Vorgabe einer Zeilennummer das Zeichen »←« und RETURN eingeben.

Falls man nach Vorgabe einer Zeilennummer die RETURN-Taste betätigt, wird die entsprechende Zeile, falls vorhanden, gelöscht. Hiermit lassen sich auch sehr schnell Programmblöcke löschen, falls man die RETURN-Taste gedrückt hält, da die Zeilenvorgabe weiterläuft und die entsprechenden Zeilennummern gelöscht werden.

Das Programm (Listing 6) geben Sie bitte mit dem MSE ein und speichern es auf Ihren Datenträger. Sie starten das Programm mit

```
SYS 49152
```

(Frank Siedel/dm)

```
 programm : auto-number      c000 c0aa
---------------------------------------

c000 : a9 0b 8d 08 03 a9 c0 8d  2f
c008 : 09 03 60 20 73 00 08 c9  9a
c010 : 5f f0 04 28 4c e7 a7 20  d0
c018 : 73 00 c9 41 d0 f5 20 73  4a
c020 : 00 18 20 6b a9 a5 14 85  c5
c028 : 26 a5 15 85 27 20 fd ae  e0
c030 : 18 20 6b a9 a5 14 85 28  ca
c038 : a5 15 85 29 a9 81 8d 02  cf
c040 : 03 a9 c0 8d 03 03 a9 80  ea
c048 : 8d 8a 02 a5 27 85 62 a5  c3
c050 : 26 85 63 a2 90 38 20 49  44
c058 : bc 20 dd bd a2 00 bd 01  76
c060 : 01 f0 09 9d 00 02 20 d2  06
c068 : ff e8 d0 f2 20 12 e1 c9  1b
c070 : 5f f0 1e c9 0d f0 2d 9d  50
c078 : 00 02 e8 20 62 a5 4c 86  49
c080 : a4 18 a5 26 65 28 85 26  58
c088 : a5 27 65 29 85 27 4c 4b  99
c090 : c0 a9 83 8d 02 03 a9 a4  e0
c098 : 8d 03 03 a9 00 8d 8a 02  37
c0a0 : 28 4c 74 a4 20 76 a5 4c  85
c0a8 : 86 a4 c1 c9 11 d0 03 4c  66
```

Listing 6. Erweitern Sie Ihr Basic um den AUTO-NUMBER-Befehl. Bitte mit dem MSE eingeben.

Angeschlossene Geräte eingeschaltet?

Dieses kleine Unterprogramm (Listing 7) dient dazu, ein Peripheriegerät (Drucker, Floppy) zuerst darauf zu testen, ob es überhaupt eingeschaltet ist. Dieses ist möglich durch Abfrage des Statusbytes. Enthält es einen anderen Wert als -128, so ist ein Fehler (DEVICE NOT PRESENT) aufgetreten. Bei nicht vorhandener Abfrage kann es schlimmstenfalls passieren, daß das Hauptprogramm abstürzt. Falls Sie diese Unterroutine in Ihr Programm einbauen wollen, so ersetzen Sie die jeweiligen PRINT-Anweisungen durch Rücksprünge in Ihr Hauptprogramm.

(dm)

```
100 REM DRUCKERTEST                        <043>
120 OPEN 4,4                               <223>
135 POKE 768,185                           <077>
155 PRINT#4:CLOSE 4                        <098>
170 POKE 768,139                           <115>
190 IF ST<>-128 THEN 205                   <170>
195 PRINT"DRUCKER NICHT EINGESCHALTET !    <063>
200 GOTO 215                               <056>
205 PRINT"DRUCKER EINGESCHALTET !          <206>
215 REM HIER GEHT'S WEITER IM PROGRAMM     <245>
225 REM FLOPPYTEST                         <047>
240 POKE 768,185                           <182>
260 OPEN 15,8,15,"I":CLOSE 15              <182>
275 POKE 768,139                           <222>
295 IF ST<>-128 THEN 310                   <004>
300 PRINT"FLOPPY NICHT EINGESCHALTET !     <062>
305 GOTO 320                               <027>
310 PRINT"FLOPPY EINGESCHALTET !           <138>
320 REM HIER GEHT'S WEITER IM PROGRAMM     <096>
```

Listing 7. Keine Probleme mehr mit ausgeschalteten Druckern oder Floppies

Zugriffszeit der Floppy verkürzen

Das vorliegende kleine Programm dient dazu, die Zugriffszeit der Floppy 1541 drastisch zu verkürzen. Der Schrittmotor, der den Schreib-Lesekopf bewegt, kann erfahrungsgemäß wesentlich schneller arbeiten, ohne daß eine sichere Funktion der Floppy gefährdet wird. Da der Schrittmotor im Interrupt bedient wird, genügt es, die Größe des Interrupt-

intervalls zu verändern, um die Drehzahl des Motors zu beeinflussen. Standardmäßig wird etwa alle 15 Millisekunden ein Interrupt ausgelöst, der den Stepper um eine Viertelspur bewegt. Durch das vorliegende Programm wird diese Zeit auf etwa 4 Millisekunden verkürzt. Alle Bewegungen des Kopfes werden dadurch etwa viermal schneller. Das hat neben der Zeitersparnis noch zwei weitere wesentliche Vorteile: Das Laufgeräusch des Kopfes wird angenehm leise und kurz, und im Falle einer Kopfjustage (MG-salvenartiges Geräusch) fährt der Kopf mit erheblich verminderter Kraft gegen den Anschlag, so daß die Gefahr einer Dejustage deutlich vermindert wird.

```
10 OPEN 1,8,15,"M-W"+CHR$(7)+
   CHR$(28)+CHR$(1)+CHR$(15)
```

(Robert Loos/dm)

Zahlensystemwandlung

Hier wollen wir einen leicht verständlichen Lösungsweg zur Umrechnung der beiden Zahlensysteme Dezimal und Hexadezimal erarbeiten.

Warum Hexadezimal?

Wir sind von Kind auf an das dezimale Zahlensystem gewöhnt. In der Computertechnik wird jedoch auch häufig das hexadezimale Zahlensystem verwendet. Warum? Um diese Frage zu beantworten, müssen wir etwas ausholen.

Wie Sie vielleicht wissen, kennt der Computer im Prinzip nur das binäre Zahlensystem, also die Kombination von 0 und 1. Aus diesen beiden Ziffern wird jede Zahl gebildet. Beispiel:

binär 1101 = dezimal 13 = hexadezimal D
binär 1100000000000000 = dezimal 49152 = hexadezimal C000

In Fachzeitschriften finden Sie oft Abkürzungen oder andere Schreibweisen, um klarzustellen, welches Zahlensystem gemeint ist. Beispiel:

binär = bin = %
hexadezimal = hex = $
dezimal = dez = #

Die Zeichen %, $ und # werden vor allem von Programmen benutzt, die irgendetwas mit Assembler (Maschinensprache) oder mit dem Betriebssystem des Computers zu tun haben, zum Beispiel Maschinensprache-Monitore. Wenn Sie mehr darüber wissen wollen, empfehle ich Ihnen das 64'er Sonderheft 8/85 (Assembler für Anfänger und Fortgeschrittene). Doch zurück zu den Zahlensystemen:

Die ersten Computer konnten nur mit binären Werten gefüttert werden, eine sehr anstrengende Sache für die Programmierer. Eine wesentliche Vereinfachung ergab das hexadezimale System. Jeweils 4 Binär-Ziffern konnten ersetzt werden durch eine einzige Hex-Ziffer. Beispiel:

bin 1101 = hex D = dez 13
bin 1000 = hex 8 = dez 8

Das liegt daran, daß das binäre Zahlensystem die Basis 2 besitzt, sich jede Zahl also in 2er-Potenzen darstellen läßt, während das hexadezimale System die Basis 16 hat. Da 2 hoch 4 den Wert 16 ergibt, können 4 Binär-Ziffern durch eine Hex-Ziffer ersetzt werden.

Unser C 64 und auch fast alle anderen Heimcomputer sind 8-Bit-Computer. Der C 64 kann 64 KByte Speicherplatz adressieren, das sind 2 hoch 16 = dezimal 65536 = hex FFFF. Sie sehen, überall taucht die 2 auf, beziehungsweise Potenzen von 2. Und das macht die Verwendung von hex-Zahlen so einfach, weil man sehr oft mit runden Hex-Zahlen zu tun hat. Beispiel: Als C 64-Besitzer werden Sie immer wieder mit der Speicherstelle 49152 zu tun haben. Mit SYS 49152 werden sehr viele Maschinenprogramme gestartet. Hexadezimal gesehen entspricht das der Zahl C000. Weitere Beispiele liefert die Speicheraufteilung des C 64:

hex	dez	Inhalt
100	256	Ende der Zeropage
400	1024	Anfang Bildschirmspeicher
800	2048	Anfang Basic-Speicher
A000	40960	8 KByte Basic-ROM
C000	49152	4 KByte RAM
D000	53248	8 KByte Interpreter-ROM

Sie sehen, daß viele wichtige Adressen »krumme« dezimale Zahlen ergeben, die sich in der Regel schlechter merken lassen. Leider kann man beim C 64 nicht auf das dezimale System verzichten. SYS-, POKE- und PEEK-Befehle lassen nur dezimale Parameter zu.

Beispiel: MSE

Ein Programm, mit dem Sie in jeder 64'er Zeitschrift und auch in diesem Sonderheft konfrontiert werden, ist der MSE. Mit dem MSE lassen sich Maschinenspracheprogramme sehr einfach, schnell und fehlerfrei abtippen. Tippen Sie ein MSE-Listing ab, haben Sie es mit Hex-Zahlen zu tun, und das aus zwei Gründen:

1. Hex-Zahlen benötigen weniger Platz als Dezimal-Zahlen, nämlich anstatt mit drei Ziffern kommt man mit zwei Ziffern aus. Das bedeutet weniger Tipp-Arbeit.

2. Wir können mehr Listing pro Seite unterbringen. Eine MSE-Zeile besteht aus der Adresse und einem Doppelpunkt (= 5 Ziffern), gefolgt von 8 Bytes plus einem Prüfbyte. Insgesamt ergibt das 23 Ziffern. Würde der MSE dezimale Zahlen verwenden, wären 33 Ziffern notwendig.

Beispiel: SMON

Selbst wenn Sie jetzt noch Anfänger sind, wird es nicht sehr lange dauern, bis Sie die ersten Schritte in Maschinensprache wagen. Spätestens dann werden Sie nicht mehr auf das hexadezimale Zahlensystem verzichten können. Denn beim SMON und auch bei den meisten anderen Maschinensprache-Monitoren wird mit Hex-Zahlen gearbeitet. Das geschieht auch hier hauptsächlich aus Platzgründen, aber auch, weil es üblich ist.

Ich hoffe, daß Sie jetzt etwas die Scheu vor diesem Zahlensystem verloren haben. Selbst wenn Sie vorerst nicht mit Hex-Zahlen arbeiten, werden Sie wissen, was gemeint ist, wenn Sie lesen: »Das Programm beginnt bei $C000 und wird mit SYS 49152 gestartet«.

1. Was ist das Hexadezimalsystem?

In unserem normalen Zahlensystem repräsentiert jede Stelle einer Zahl eine Zehnerpotenz. Ein Beispiel: Die Zahl 4714 läßt sich auch als Summe von Zehnerpotenzen schreiben.

$4714 = 4*10^3 + 7*10^2 + 1*10^1 + 4*10^0 = 4*1000 + 7*100 + 1*10 + 4*1$

Beim Hexadezimalsystem wird nun jede Stelle einer Zahl nicht mehr durch eine Zehner-, sondern durch eine Sechzehnerpotenz repräsentiert. Auch hier wieder ein Beispiel: Die Hexadezimalzahl 0324 bedeutet nichts anderes als $3*16^2 + 2*16^1 + 4*16^0$ (=3*256+2*16+4*1).

Dies hat aber noch weitere Konsequenzen:

Im Dezimalsystem wird eine Stelle immer von 0 bis 9 (insgesamt 10 Ziffern) durchgezählt, bevor die nächste Stelle um eins erhöht wird. Also

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14...

Hex, Dez, Bin . . .?

Im Hexadezimalsystem jedoch wird eine Stelle um sechzehn Werte erhöht, bevor zur nächsten Stelle ein Wert hinzugefügt wird. Da aber unsere Ziffern von 0 bis 9 dazu nicht ausreichen, wurden zusätzlich die Buchstaben A bis F herangezogen. Sie vertreten die Zahlenwerte 10 bis 15 (von 0 bis 15 sind es 16 Werte!). Es bedeuten:

A = 10
B = 11
C = 12
D = 13
E = 14
F = 15

 0 1 2 3 4 5 6 7 8 9 A B C D E F
10 11 12 13 14... 1A 1B 1C...

Die Dezimalzahl 10 ist also gleichwertig mit dem Hexadezimalwert A. Damit wären wir auch schon bei der Umrechnung.

2. Dezimal – Hexadezimal

Wenn wir eine Dezimalzahl in Hexadezimal umrechnen wollen, so bauen wir den Hex-Wert Stelle für Stelle von links nach rechts auf.

Nehmen wir also an, wir möchten die Dezimalzahl 41717 in Hexadezimal umrechnen. Dazu teilen wir sie erst einmal durch 16^3

$41717 : 16^3 = 10{,}1848145$

Uns interessiert hier nur die Vorkommastelle 10. Sie ist gleichbedeutend mit dem Hex-Wert **A**. Er bildet die letzte Stelle unserer Hexadezimalzahl.

Nun müssen wir von unserer Dezimalzahl $10 * 16^3$ abziehen.

Also $41717 - 10 * 16^3 = 757$

Um die nächsten Stellen unserer Hex-Zahl zu erhalten, führen wir diese Prozedur nun noch mit 16^2 und 16^1 durch:

$757 : 16^2 = 2{,}9703125$ **(=2)**
$757 - 2 * 16^2 = 245$
$245 : 16^1 = 15{,}3125$ **(=F)**
$245 - 15 * 16^1 =$ **5**

Als endgültige Umrechnung der Zahl 41717 ins Hexadezimalsystem erhalten wir also **A2F5**.

3. Hexadezimal – Dezimal

Diese Umrechnung ist schon wesentlich einfacher. Um die Hex-Zahl A2F5 wieder zurückzurechnen, geht man wie folgt vor:

$A * 16^3 + 2 * 16^2 + F * 16^1 + 5 * 16^0$

Da man aber mit den Buchstaben A und F nicht rechnen kann, müssen diese als Dezimalzahlen angegeben werden.

$10 * 16^3 + 2 * 16^2 + 15 * 16^1 + 5 * 16^0$

Wenn Sie dies auf Ihrem C 64 einmal ausrechnen, so werden Sie als Ergebnis wieder die Zahl 41717 erhalten!

Als Abschluß unseres kleinen Kurses könnten Sie einmal versuchen, ein Basic-Programm zu schreiben, das diese Berechnungen ausführt.

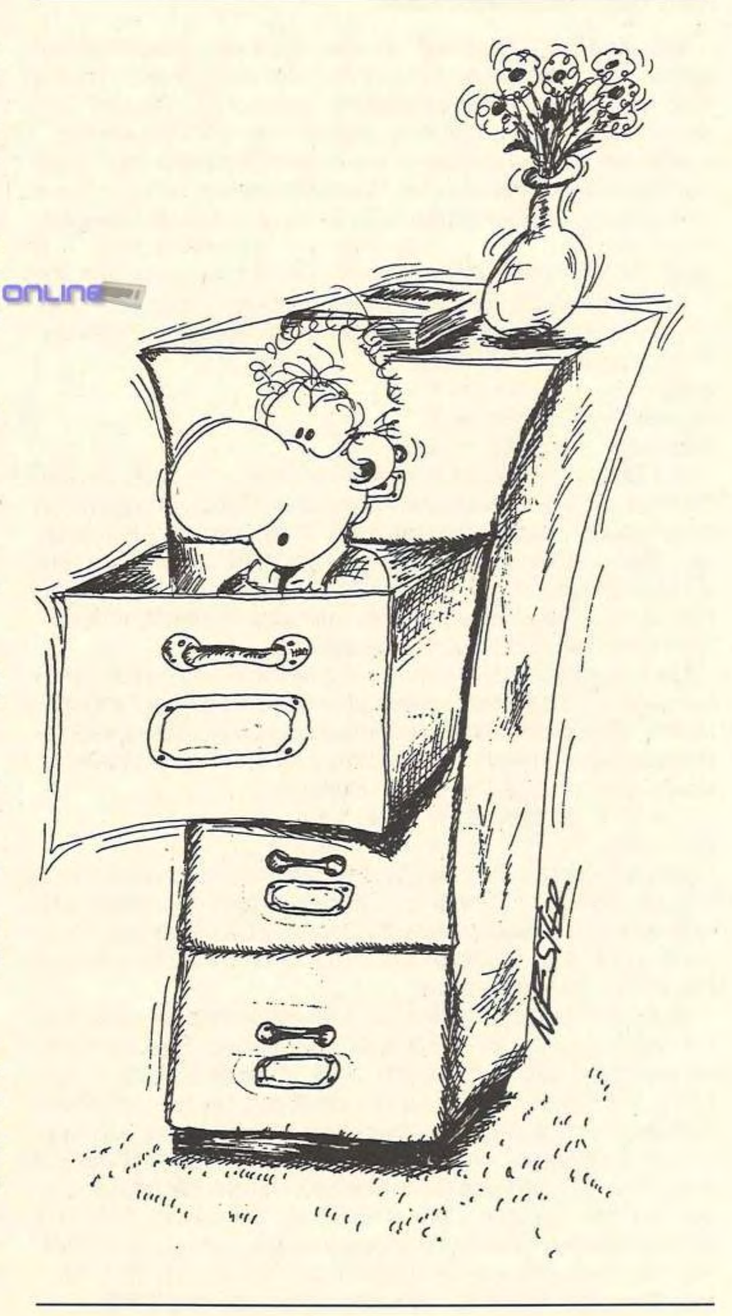
Kleine Merkhilfe

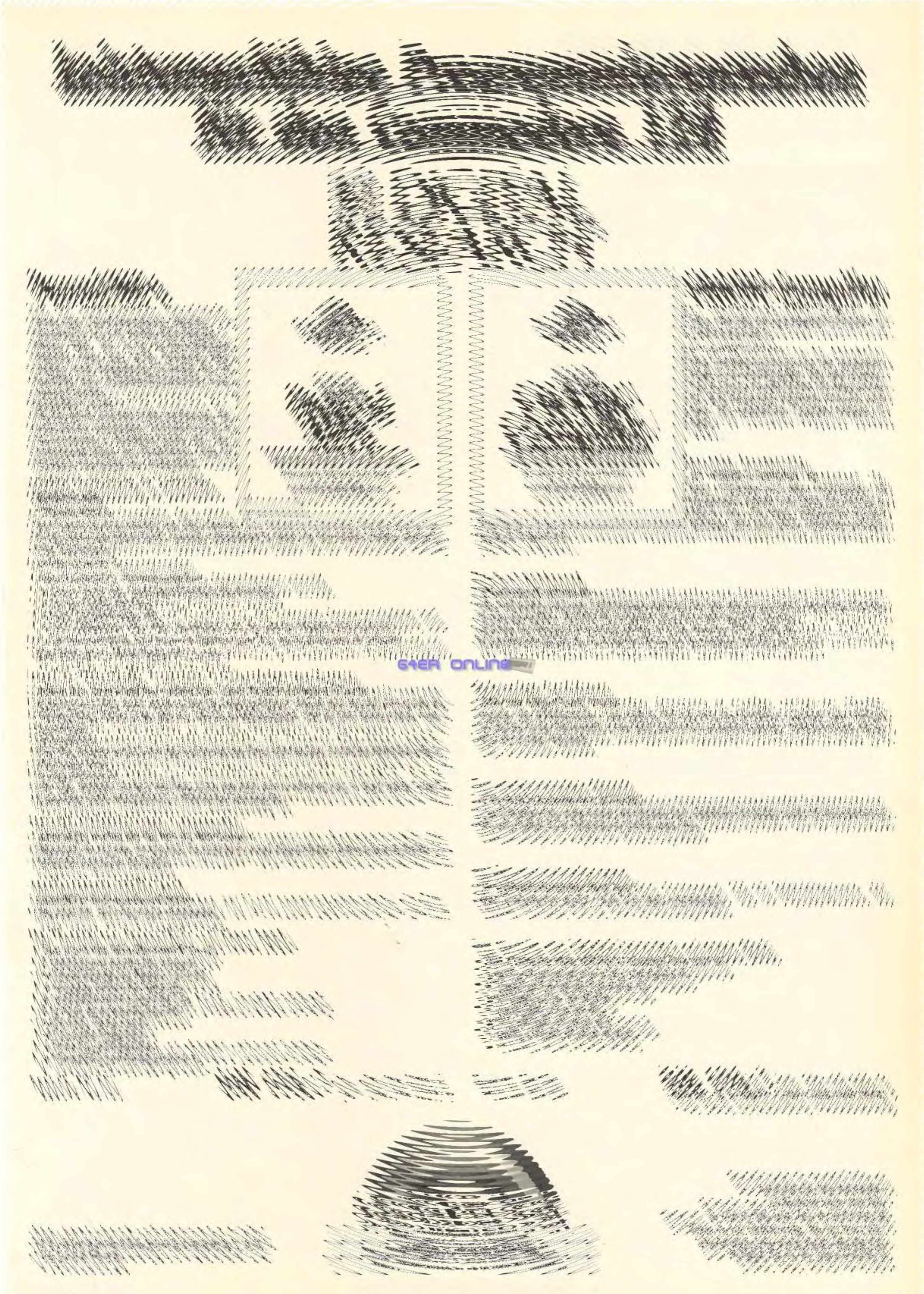
Gerade wenn man anfängt, sich mit dem hexadezimalen Zahlensystem zu beschäftigen (und dafür gibt es einige gute Gründe), ist man froh über ein paar kleine Eselsbrücken. Am Anfang kommt man immer wieder ins Stolpern, weil man sich nicht so schnell an den Dezimalwert von zum Beispiel Hex C erinnern kann. Dies geht wesentlich leichter, wenn man die Anfangsbuchstaben der beiden Systeme gegenüberstellt.

C = **Z**wölf
D = **D**reizehn
F = **F**ünfzehn

Dezimal	Hexadezimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15
10	16
11	17
12	18
4096	1000
49152	C000 = 12 * 4096

Tabelle 3. Einige Beispiel, wie Hex-Werte in dezimal und umgekehrt aussehen würden. Man sieht deutlich die unterschiedliche Darstellungsweise der Zahlensysteme.





Zeichen in vierfacher Größe

Ein tolles Programm zur Darstellung des normalen Zeichensatzes in vierfacher Breite und Höhe. »Blow Up« ist gedacht für große Überschriften auf dem Bildschirm, für Nummerierungen innerhalb von mehrstufigen Menü-Programmen und die hervorgehobene Ausgabe von Ergebnissen. Der Trick dabei besteht darin, daß die Zeichen aus normalen Blockgrafik-Elementen zusammengesetzt werden. Dadurch erspart man die Umstellung auf den Grafikmodus und auf spezielle Zeichensätze.

Bitte geben Sie zuerst das Maschinenprogramm »Blow Up« (Listing 8) mit Hilfe des MSE ein und speichern es. Das Demo-Programm (Listing 9) können Sie ebenfalls abtippen, müssen aber nicht. Nachdem Sie es ebenfalls gespeichert haben, starten Sie es mit RUN.

Zur Bedienung: Das Programm ist mit LOAD″BLOW UP″,8,1 zu laden. Die Übergabe des auszudruckenden Textes erfolgt mit

`SYS 51968($)`

Dabei kann eine Stringvariable ($) in der Form A$ oder ″TEST″ übergeben werden, zum Beispiel SYS 51968 (″TEST″). Auch kombinierte Strings (A$+B$) sind möglich. Der Ausdruck muß jedoch in Klammern stehen, da sonst die Fehlermeldung SYNTAX ERROR ausgegeben wird. Die verschiedenen Zeichensätze des C64 werden über einen POKE-Befehl erreicht. Dabei gilt:

`POKE 52073,208`
- Großschrift/Blockgrafik (Grundeinstellung)

`POKE 52073,212`
- Großschrift/Blockgrafik invertiert

`POKE 52073,216`
- Groß-/Kleinschrift

`POKE 52073,220`
- Groß-/Kleinschrift invertiert

Steuerzeichen innerhalb eines Strings werden ähnlich wie beim LISTen dargestellt, aber nicht ausgeführt. Die Eingabe eines Leerstrings führt zur Ausgabe eines 255 Zeichen langen Bereichs aus dem Basic-Speicher. Dieser »Fehler« wurde nicht beseitigt, da diese wirren Zeichen ganz reizvoll sein können.

Die Ausgabe der vergrößerten Zeichen beginnt ab der Position des Cursors. Die Cursor-Position wird durch »Blow Up« nicht verändert. Zeichen, die im unteren Bildschirmbereich nicht mehr vollständig abgebildet werden können, wandern an den Bildschirmanfang. Da Strings bis zu 255 Zeichen lang sein können, kann der Bildschirm von einem String mehrfach überschrieben werden.

(Ludwig Wolff/dm)

```
PROGRAMM : BLOW UP              CB00 CBFC
------------------------------------------

CB00 : 20 FA AE 20 9E AD 20 A3   6C
CB08 : B6 86 28 84 29 8D EA CB   DE
CB10 : AD 0E DC 29 FE 8D 0E DC   6F
CB18 : 18 A5 D1 65 D3 AA A5 D2   F3
CB20 : 69 00 A8 86 FA 86 71 98   5F
CB28 : 85 FB 49 DC 85 72 A9 0A   3F
CB30 : 8D EB CB A0 04 A2 00 86   1C
CB38 : FF 86 02 A5 FB C9 07 90   FB
CB40 : 06 A5 FA C9 6D B0 DC A1   24
CB48 : 28 30 0C C9 40 90 0C 29   D0
CB50 : 3F C9 40 90 06 29 1F 29   0E
CB58 : 7F 09 40 18 0A 26 FF 0A   55
CB60 : 26 FF 0A 26 FF 85 FE 18   26
CB68 : A9 D0 65 FF 85 FF A5 01   C3
CB70 : 29 FB 85 01 A2 00 A1 FE   C7
CB78 : 85 FC E6 FE A1 FE 85 FD   39
CB80 : E6 FE A5 01 09 04 85 01   38
```

Listing 8. Zaubern Sie viermal so große Buchstaben auf Ihren Bildschirm. Bitte mit dem MSE eingeben.

```
CB88 : 66 FC 2A 66 FC 2A 66 FD   7A
CB90 : 2A 66 FD 2A 29 0F AA 88   79
CB98 : BD EC CB 91 FA AD 86 02   2C
CBA0 : 91 71 C4 02 D0 E2 98 18   12
CBA8 : 69 28 85 02 69 04 A8 C9   B4
CBB0 : A4 D0 BB CE EB CB D0 09   F7
CBB8 : A9 0A 8D EB CB A9 7C D0   E5
CBC0 : 02 A9 04 18 65 FA 90 04   13
CBC8 : E6 FB E6 72 85 FA 85 71   DD
CBD0 : E6 28 D0 02 E6 29 CE EA   07
CBD8 : CB F0 03 4C 33 CB AD 0E   CA
CBE0 : DC 09 01 8D 0E DC 20 F7   6B
CBE8 : AE 60 00 04 20 7B 6C 62   9B
CBF0 : 7E 61 7F FC 7C FF E1 FE   EB
CBF8 : E2 EC FB A0 00 00 00 00   63
```

Listing 8. Schluß

```
100 REM *********************************    <238>
101 REM *     DEMO + BOOT  PROGRAMM     *    <061>
102 REM *             FUER              *    <167>
103 REM *                               *    <152>
104 REM *         B L O W   U P         *    <191>
105 REM *                               *    <154>
106 REM *  EINE BASIC-ZUSATZFUNKTION    *    <148>
107 REM *   FUER ZEICHENDARSTELLUNG     *    <120>
108 REM *  IN 4-FACHER HOEHE + BREITE   *    <149>
109 REM *                               *    <158>
110 REM *             VON               *    <243>
111 REM *                               *    <160>
112 REM *         LUDWIG WOLFF          *    <009>
113 REM *       NIEDERHUTSTR.42         *    <208>
114 REM *        5483 AHRWEILER         *    <045>
115 REM *       TEL.02641/34867         *    <213>
116 REM *                               *    <165>
117 REM *          15.11.1985           *    <173>
118 REM *********************************    <000>
200 :                                        <176>
300 :                                        <022>
400 :                                        <122>
500 IF A=0 THEN A=1:LOAD"BLOW UP",8,1        <099>
520 C=646:POKE 53280,0:POKE 53281,0          <221>
530 Z1$="@ABCDEFGHIJKLMNOPQRSTUVWXYZ £↑123
    4567890"                                 <112>
540 Z2$="ZVXSHQ*M¥X-+LFGJQYR/£RNB*U+-Z↑UYT
    @RFCDT UYT GHJK£LNDVD B+[]←"             <222>
600 GOSUB 1000:S=9:Z=6:GOSUB 1100            <150>
610 PRINT"EIN DEMO-PROGRAMM FUER"            <118>
620 S=12:Z=11:GOSUB 1100:PRINT" B L O W{3S
    PACE}U P":T=600:GOSUB 1200               <221>
650 GOSUB 1000:PRINT"{HOME,2DOWN,2RIGHT,SP
    ACE}BLOW UP{2SPACE}IST EIN HILFSPROGRA
    MM ZUR"                                  <033>
660 PRINT"{DOWN,2RIGHT}DARSTELLUNG VON ZEI
    CHEN AUS DEM C64-"                       <136>
670 PRINT"{2DOWN,4RIGHT}ZEICHENSATZ IN 4-F
    ACHER GROESSE:":T=300:GOSUB 1200         <159>
680 S=6:Z=11:POKE C,1:GOSUB 1100:SYS 51968
    ("BLOW UP"):T=150:GOSUB 1200             <124>
690 S=4:Z=16:POKE C,10:GOSUB 1100:SYS 5196
    8("DEMO-PRG")                            <052>
700 T=400:GOSUB 1200                         <083>
710 GOSUB 1000:S=11:Z=13:GOSUB 1100:PRINT"
    DAS KANN BLOW UP:"                       <150>
720 T=150:GOSUB 1200:PRINT"{CYAN,HOME}":SY
    S 51968(Z1$)                             <035>
730 S=13:Z=19:GOSUB 1100:PRINT"{YELLOW}GRO
    SSE{2SPACE}ZEICHEN."                     <065>
740 T=450:GOSUB 1200:POKE 52073,216:PRINT"
    {LIG.RED,HOME}":SYS 51968(Z1$)           <170>
750 S=13:Z=19:GOSUB 1100:PRINT"{YELLOW}KLE
    INE":GOSUB 1200:GOSUB 1000               <058>
760 S=7:Z=2:POKE C,10:GOSUB 1100:SYS 51968
    ("GROSSE,")                              <020>
770 S=8:Z=6:POKE C,14:GOSUB 1100:SYS 51968
    ("KLEINE")                               <017>
780 S=0:Z=10:POKE C,7:GOSUB 1100:SYS 51968
    ("BUCHSTABEN")                           <042>
790 GOSUB 1200:GOSUB 1000:S=10:Z=10:GOSUB
    1100:PRINT"UND DIE BLOCKGRAHIK:"         <042>
800 T=150:GOSUB 1200:PRINT"{HOME}";:POKE 5
    2073,208                                 <059>
810 SYS 51968(Z2$)                           <141>
820 GOSUB 1200:POKE 52073,212                <039>
830 SYS 51968(Z2$)                           <161>
```

Listing 9. Ein Demo-Programm zu »BLOW UP«

```
840 GOSUB 1200:GOSUB 1010:POKE 52073,216:S
    =13:Z=7:POKE C,14:GOSUB 1100              <222>
850 SYS 51968("AUCH"):S=8:Z=13:GOSUB 1100:
    SYS 51968("INVERS")                        <068>
860 GOSUB 1200:GOSUB 1000:S=11:Z=7:GOSUB 1
    100:PRINT"{YELLOW}UND HIER EIN TRICK:"    <216>
870 L$="{20SPACE}"                             <181>
880 PRINT SPC(90)L$SPC(20)L$"{LIG.RED}"SPC
    (20)L$SPC(20)"{LIG.BLUE}"L$                <181>
890 POKE 52128,165:S=10:Z=10:GOSUB 1100:SY
    S 51968("64'ER")                           <030>
900 POKE 52128,145:POKE 52073,208:T=350:GO
    SUB 1200:GOTO 600                          <187>
999 STOP                                       <047>
1000 REM -- BILDSCHIRM MASKE --                <236>
1005 PRINT"{CLR}"                              <231>
1010 PRINT"{HOME,YELLOW}R*********************
     ************************S";               <071>
1020 FOR I=0 TO 20:PRINT"="SPC(38)"=";:NEX
     T                                         <155>
1030 PRINT"= TASTE DRUECKEN ODER WARTEN !"
     SPC(9)"=";                                <239>
1040 PRINT"Z*********************************
     *********X";                              <195>
1050 PRINT" * BLOW UP * L.WOLFF * AHRWEILE
     R 1985 *";:RETURN                         <184>
1100 REM -- CRSR SET --                        <152>
1110 POKE 211,S:POKE 214,Z:SYS 58640:RETUR
     N                                         <038>
1200 REM -- GET --                             <027>
1210 I=0                                       <127>
1220 I=I+1:IF I>T THEN RETURN                  <168>
1230 GET A$:IF A$=""THEN 1220                  <008>
1240 RETURN                                    <026>
```

Listing 9. »BLOW-UP« (Schluß)

Formatierte Zahlenausgabe

Wie allgemein bekannt und vielfach bemängelt, bietet das Commodore-Basic keinen PRINT USING-Befehl. Doch gerade bei kommerziellen Problemen kann auf eine Druckaufbereitung von Zahlen nicht verzichtet werden. Eine Rechnung, in der nicht einmal die Dezimalpunkte untereinander stehen, ist eben unübersichtlich und keine Reklame. Für die Druckaufbereitung gibt es verschiedene Lösungen.

Die beste Lösung ist ein Maschinenprogramm. Doch zuerst stellt sich die Frage, wie ein Maschinenprogramm für die Druckaufbereitung aufgerufen werden soll. Offensichtlich ist die USR-Funktion am geeignetsten, da sie sowohl in einer PRINT- als auch in einer PRINT#- oder LET-Anweisung verwendet werden kann. Die Funktion wird aufgerufen durch USR(X),L,NK.

Dabei ist X die Zahl, die aufbereitet werden soll,
L die Gesamtfeldlänge der aufbereiteten Zahl einschließlich Vorzeichen und Dezimalkomma und
NK die Anzahl der darzustellenden Nachkommastellen.

Die USR-Funktion wandelt zunächst die Zahl X in einen ASCII-String um und berechnet die Stringlänge und die Anzahl der Nachkommastellen. Wenn bei der Wandlung von X der Interpreter die Exponentialform wählt, dann wird die Exponentialdarstellung zunächst in die Fließkommadarstellung umgewandelt. Danach folgt die Aufbereitung der Nachkommastellen. Fehlende Nachkommastellen werden durch angehängte Nullen ergänzt. Müssen Nachkommastellen abgeschnitten werden, dann rundet der Interpreter die Zahl, wenn die erste abgeschnittene Dezimalstelle größer oder gleich 5 ist. Wenn die Anzahl der gewünschten Nachkommastellen null ist, wird die Zahl als ganze Zahl (integer) ohne Dezimalkomma aufbereitet. Wegen der kaufmännischen Anwendung ersetzt die Routine den Dezimalpunkt durch ein Dezimalkomma. Nach der Aufbereitung der Nachkommastellen erhält der String durch Voranstellen von Leerzeichen die erforderliche Länge. Ist der String nach der Aufbereitung der Nachkommastellen schon länger als gewünscht, so wird er nicht mehr verändert, sondern in voller Länge ausgegeben, um einen Datenverlust zu verhindern.

Das hier vorgestellte Maschinenprogramm kann vom Anwender in den Speicherbereich geladen werden, der ihm am geeignetsten erscheint. Wenn druckaufbereitete Werte einer Variablen zugewiesen werden sollen, dann muß die Anweisung

```
A$=""+USR(X),L,NK
```

oder

```
A$=(USR(X),L,NK)+""
```

lauten, da dann das Ergebnis der Stringverknüpfung in den Speicherbereich kopiert wird und der Variablen A$ dauerhaft zugewiesen ist.

Vor dem ersten Aufruf der USR-Funktion muß jetzt noch in Adresse 785 (Low-Byte) und 786 (High-Byte) die Startadresse der USR-Funktion hinterlegt werden. Listing 10 zeigt das Basic-Ladeprogramm für die USR-Funktion. Die Ladeadresse können Sie selbst bestimmen. Das Ladeprogramm setzt die Startadresse der USR-Funktion in den Speicherstellen 785 und 786 entsprechend. Zur Verdeutlichung der Anwendung der USR-Funktion enthält das Ladeprogramm verschiedene Druckaufbereitungen der Zahl. Das Ergebnis des Beispiels ist in Bild 2 wiedergegeben.

(Dr. Michael Irskens/dm)

```
100 S=0:INPUT "STARTADRESSE";B                 <182>
110 FOR I=B TO B+335                           <000>
120 READ A                                     <160>
130 S=S+A:POKE I,A                             <062>
140 NEXT I                                     <224>
150 IF S<>36986 THEN PRINT "{RVSON,SPACE}F
    EHLER IN DEN DATAZEILEN{SPACE,RVOFF}":
    STOP                                       <119>
160 POKE 785,B-256*INT(B/256):POKE 786,B/2
    56                                         <040>
170 PRINT " ALLES OK. "                        <036>
190 REM TESTBEISPIELE                          <050>
200 FOR E=0 TO 9                               <238>
210 F=±*10↑E                                   <176>
220 A$=(USR(F),11,0)+""                        <178>
230 PRINT A$;USR(F),13,1;USR(F),14,2           <024>
240 NEXT E                                     <036>
250 END                                        <252>
299 REM DATAZEILEN                             <190>
300 DATA 32,141,173,32,221,189,32,253,174,
    32,158,183,134,88,32,253,174,32            <212>
301 DATA 158,183,134,87,104,104,162,255,16
    0,0,232,189,0,1,240,117,201,69,240         <057>
302 DATA 8,201,46,208,242,138,168,208,238,
    173,2,1,201,46,208,12,202,160,1            <250>
303 DATA 200,185,1,1,153,0,1,208,247,189,2
    ,1,41,15,10,133,2,10,10,101,2,125          <139>
304 DATA 3,1,233,47,188,1,1,192,45,240,23,
    105,3,134,2,229,2,168,169,48,157           <106>
305 DATA 0,1,232,136,208,249,169,0,157,0,1
    ,240,168,133,2,169,0,157,0,1,138           <207>
306 DATA 24,101,2,168,189,0,1,240,8,201,48
    ,176,4,169,48,208,1,202,153,0,1            <048>
307 DATA 136,208,236,169,46,141,1,1,208,12
    9,152,240,18,165,87,208,7,152,170          <211>
308 DATA 189,1,1,208,119,169,44,153,0,1,20
    8,12,196,87,240,40,169,44,157,0            <159>
309 DATA 1,232,208,16,132,2,56,138,229,2,5
    6,233,1,197,87,240,19,176,72,168           <011>
310 DATA 169,48,157,0,1,232,200,196,87,208
    ,247,169,0,157,0,1,173,1,1,201,48          <115>
311 DATA 176,17,232,138,168,185,255,0,153,
    0,1,136,208,247,169,48,141,1,1,228         <091>
312 DATA 88,176,20,164,88,189,0,1,153,0,1,
    136,202,16,246,169,32,153,0,1,136          <163>
313 DATA 16,250,169,0,160,1,76,135,180,56,
    229,87,133,2,138,56,229,2,170,189          <206>
314 DATA 0,1,201,53,144,179,138,168,136,24
    0,24,185,0,1,201,48,144,246,24,105         <086>
315 DATA 1,201,58,153,0,1,208,157,169,48,1
    53,0,1,208,229,138,168,185,0,1,153         <028>
316 DATA 1,1,136,208,247,169,49,141,1,1,23
    2,208,131                                  <030>
```

Listing 10. Formatierte Zahlenausgabe auf Ihrem Drucker – nun kein Problem mehr



3	3,1	3,14
31	31,4	31,42
314	314,2	314,16
3142	3141,6	3141,59
31416	31415,9	31415,93
314159	314159,3	314159,27
3141593	3141592,7	3141592,65
31415927	31415926,5	31415926,50
314159265	314159265,0	314159265,00
3141592650	3141592650,0	3141592650,00

Bild 2. Beispiele für verschiedene Druckaufbereitungen

Directory sortieren

Ist Ihnen das auch schon öfters passiert: Sie hatten auf einer Diskette ein schön geordnetes Directory und beim Speichern eines weiteren Files stand dieses nicht sauber am Schluß des Directorys, sondern mitten zwischen den anderen Programmen. Mit dem hier vorgestellten Sortierprogramm (Listing 11) ist es nun möglich, diese »falsch hineingeratenen« Files herauszunehmen und an geeigneter Stelle wieder einzusetzen. Auch das Einsetzen von Trennstrichen ist möglich. Bild 3 und Bild 4 zeigen Ihnen den Unterschied zwischen einem unsortierten zu einem sortierten Directory. Mit etwas Geschick kann man seine Disketten auch so umsortieren, daß an erster Stelle immer ein Ladeprogramm für das Hauptprogramm steht. Man kann sich dadurch viel Sucharbeit sparen, denn man lädt einfach das erste Programm.

Nach dem Starten des Programms legt man die Diskette ein, die sortiert werden soll und drückt eine Taste. In der linken oberen Ecke wird nun die Sektornummer des Directory-Blocks angezeigt, den der Computer gerade einliest, rechts daneben die Anzahl der in den Speicher eingelesenen Files.

Hinweise zum Abtippen
Im folgenden Listing tauchen eventuell unterstrichene oder überstrichene Zeichen auf. Diese sind folgendermaßen einzugeben:
unterstrichen: SHIFT-Taste und Buchstabe
überstrichen: COMMODORE-Taste und Buchstabe
Bei Begriffen in geschweiften Klammern, zum Beispiel {CLR}, muß die Taste SHIFT und CLR gedrückt werden und weder die Klammer noch das Wort CLR.
Die <Zahl> am Ende jeder Basic-Zeile darf nicht eingegeben werden.
Genaueres steht im Beitrag Checksummer 64 auf Seite 135.

```
100 GOTO 200                                    <036>
110 INPUT#1,F1,F$,F2,F3:IF F1=0 THEN RETUR
    N                                           <139>
120 PRINT F1;F$,F2;F3:END                       <030>
130 IF NO+EN>=C THEN RETURN                     <043>
131 SYS 53056,0,2,2,23,18:NO=NO+1               <007>
135 POKE 214,23:POKE 211,2:SYS 58732:PRINT
     NA$(ZU(NO+EN-1));:RETURN                   <057>
140 IF NO=0 THEN RETURN                         <227>
141 SYS 53056,1,2,2,23,18:NO=NO-1               <023>
145 POKE 214,2:POKE 211,2:SYS 58732:PRINT
    NA$(ZU(NO+1)):RETURN                        <250>
200 POKE 53280,6:POKE 53281,6                   <108>
201 PRINT"{CLR,YELLOW,RVSON}"TAB(9)"EPLOX'
    S DIRECTORY-SORT"                           <125>
202 PRINT TAB(8)"------------------------"
    :IF DG=1 THEN 210                           <002>
203 PRINT" F1{2SPACE}EINTRAG MARKIEREN/EIN
    ORDNEN"                                     <009>
204 PRINT" F2{2SPACE}TRENNSTRICH ERZEUGEN" <081>
205 PRINT" F3{2SPACE}AUF"                       <146>
206 PRINT" F4{2SPACE}LOESCHEN"                  <200>
207 PRINT" F5{2SPACE}AB"                        <227>
208 PRINT" F8{2SPACE}SPEICHERN"                 <022>
209 PRINT "{3SPACE,3DOWN}BITTE DISKETTE EI
    NLEGEN":DG=1:POKE 198,0:WAIT 198,63:GO
    TO 201                                      <027>
210 OPEN 1,8,15,"I":DIM AN$(145),NA$(145),
    RE$(145),ZU(146),SN(19)                     <244>
215 FOR I=1 TO 18:READ SN(I):NEXT               <171>
218 FOR I=52992 TO 53242:READ X:POKE I,X:N
    EXT                                         <045>
219 GOSUB 110:OPEN 2,8,2,"#":GOSUB 110          <010>
220 S=1:C=1:N$=CHR$(0):NN$=N$+N$+N$             <157>
223 FOR I=1 TO 10:NU$=NU$+NN$:NEXT              <234>
225 AN$(0)=CHR$(128)+CHR$(18)+CHR$(1):NA$(
    0)="----------------"                       <176>
226 RE$(0)=NN$+NN$+NN$+N$+N$                    <003>
230 PRINT#1,"U1 2 0 18"S:PRINT"{HOME,LIG.R
    ED}"S"{LEFT,SPACE}"                         <101>
235 GET#2,T$:GET#2,S$:S=ASC(S$+N$)              <089>
240 FOR BP=0 TO 7:PRINT#1,"B-P 2";BP*32+2  <217>
270 SYS 52992,2,3,X$:AN$(C)=X$:IF LEFT$(X$
    ,1)=N$THEN NEXT:GOTO 320                    <049>
290 SYS 52992,2,16,X$:NA$(C)=X$:SYS 52992,
    2,11,X$:RE$(C)=X$                           <036>
300 ZU(C)=C:PRINT"{HOME,4RIGHT}"C"{LEFT,SP
    ACE}":C=C+1:NEXT                            <206>
320 IF T$<>""THEN 230                           <028>
400 CLOSE 2:POKE 650,128:PRINT"{HOME,2DOWN
    ,BLACK}>{CYAN,UP}";:CP=2:NO=0:EN=C:IF
    EN>23 THEN EN=23                            <033>
410 PRINT CHR$(13)TAB(2)NA$(NU+1);:NU=NU+1
    :IF NU<EN-1 THEN 410                        <046>
440 GET TA$:IF TA$=""THEN 440                   <197>
450 IF TA$="{F3}"THEN IF CP>2 THEN CP=CP-1
    :SYS 53056,0,2,0,23,0                       <077>
455 IF TA$="{F3}"THEN IF CP=2 THEN GOSUB 1
    40                                          <223>
460 IF TA$="{F5}"THEN IF CP<EN THEN CP=CP+
    1:SYS 53056,1,2,0,23,0                      <027>
465 IF TA$="{F5}"THEN IF CP>=23 THEN GOSUB
     130                                        <231>
470 IF TA$="{F1}"THEN 1000                      <002>
480 IF TA$="{F2}"THEN IF C<145 THEN F=1:C=
    C+1:PRINT"{LIG.RED,HOME,4RIGHT}"C-1"{L
    EFT,SPACE,CYAN}":EN=EN+1:IF EN>23 THEN
     EN=23                                      <128>
485 IF TA$="{F2}"THEN IF F=1 THEN F=0:TE=0
    :TE$=NA$(0):GOTO 1010                       <150>
490 IF TA$="{F8}"THEN 3000                      <045>
495 IF TA$="{HOME}"THEN 3040                    <057>
500 GOTO 440                                    <006>
1000 TE=ZU(NO+CP-1):TE$=NA$(TE)                 <238>
1005 SYS 53056,0,CP,2,23,18:IF C>23 THEN E
     N=EN+1:GOSUB 135:EN=EN-1                   <173>
1007 FOR I=NO+CP-2 TO C-1:ZU(I+1)=ZU(I+2):
     NEXT                                       <072>
1010 POKE 214,CP:POKE 211,19:SYS 58732          <217>
1020 PRINT"TT"TE$                               <102>
1040 GET TA$:IF TA$=""THEN 1040                 <101>
1050 IF TA$="{F3}"THEN IF CP>2 THEN CP=CP-
     1:SYS 53056,0,2,19,24,37:SYS 53056,0,
     2,0,23,0                                   <235>
1055 IF TA$="{F3}"THEN IF CP=2 THEN GOSUB
     140                                        <061>
1060 IF TA$="{F5}"THEN IF CP<EN THEN CP=CP
     +1:SYS 53056,1,2,19,24,37:SYS 53056,1
     ,2,0,23,0                                  <033>
1065 IF TA$="{F5}"THEN IF CP=23 THEN GOSUB
      130                                       <152>
1070 IF TA$="{F1}"THEN 2000                     <104>
1075 IF TA$="{F4}"THEN TE$="{16RIGHT}":C=C
     -1:EN=C:IF EN>23 THEN EN=23                <201>
1076 IF TA$="{F4}"THEN PRINT"{HOME,LIG.RED
     ,4RIGHT}"C-1"{LEFT,SPACE,CYAN}":GOTO
     2050                                       <027>
1080 GOTO 1040                                  <132>
2000 SYS 53056,1,CP,2,23,18                     <122>
2040 FOR I=C-1 TO NO+CP-1 STEP-1:ZU(I+1)=Z
     U(I):NEXT:ZU(NO+CP-1)=TE                   <093>
2050 POKE 214,CP:POKE 211,2:SYS 58732           <199>
2060 PRINT TE$"{19SPACE}"                       <149>
2199 GOTO 440                                   <183>
3000 OPEN 2,8,2,"#":T=18:FOR I=0 TO INT((C
     -2)/8):IF 8*I+8>=C-1 THEN T=0              <147>
3010 PRINT#1,"B-P 2 0":PRINT#2,CHR$(T)CHR$
     (SN(I+2))                                  <165>
3020 FOR BP=0 TO 7:PRINT#1,"B-P 2";BP*32+2 <203>
3023 IF BP+8*I+1>=C THEN PRINT#2,NU$;:NEXT
```

Listing 11. Komfortables Sortieren Ihres Directory.

```
     :GOTO 3030                                 <082>
3025 PRINT#2,AN$(ZU(BP+8*I+1))NA$(ZU(BP+8*
     I+1))RE$(ZU(BP+8*I+1));:NEXT               <040>
3030 PRINT#1,"U2 2 0 18"SN(I+1):PRINT"{LIG
     .RED,HOME}"TAB(36)SN(I+1)"{LEFT,SPACE
     ,CYAN}":GOSUB 110:NEXT                     <032>
3040 CLOSE 1:CLOSE 2:RUN                        <235>
10000 DATA 1,4,7,10,13,16,2,5,8,11,14,17,3
      ,6,9,12,15,18                             <072>
10001 DATA 32,253,174,32,158,183,32,30,225
      ,32,253,174,32,158,183,138,72,32,253 <099>
10002 DATA 174,32,139,176,133,73,132,74,32
      ,163,182,104,32,117,180,160,2,185         <074>
10003 DATA 97,0,145,73,136,16,248,200,32,1
      8,225,145,98,200,196,97,208,246,32        <169>
10004 DATA 204,255,96,0,0,0,0,32,245,207
      ,138,72,32,245,207,224,0,176,3,76         <032>
10005 DATA 72,178,224,24,176,249,134,251,3
      2,245,207,224,0,144,240,224,39,176        <109>
10006 DATA 236,134,253,32,245,207,224,25,1
      76,227,134,252,232,138,56,229,251         <137>
10007 DATA 144,218,240,216,133,250,32,245,
      207,224,40,176,207,228,253,144,203        <197>
10008 DATA 134,254,104,170,165,172,72,165,
      173,72,165,174,72,165,175,72,224,0        <108>
10009 DATA 208,22,166,251,198,250,240,44,3
      2,240,233,232,189,240,236,133,172         <223>
10010 DATA 181,217,32,219,207,48,236,202,2
      40,3,76,72,178,166,252,198,250,240        <054>
10011 DATA 16,32,240,233,202,189,240,236,1
      33,172,181,217,32,219,207,48,236,164 <107>
10012 DATA 254,32,240,233,32,36,234,169,32
      ,145,209,136,196,253,16,249,76,88         <232>
10013 DATA 233,41,3,13,136,2,133,173,32,22
      4,233,164,254,177,172,145,209,177         <205>
10014 DATA 174,145,243,136,196,253,16,243,
      96,32,253,174,76,158,183                  <199>
```

Listing 11. Schluß

```
0 "64'ER 11/85        " 64'ER
16    "1. PRG"              PRG
7     "MSE V1.0"            PRG
58    "APFELMANN"           PRG
13    "HARDCOPY KOALA"      PRG
12    "2. PRG"              PRG
69    "PROFIPRINT"          PRG
4     "APFELROUTINEN"       PRG
69    "LYRIC 3.0"           PRG
12    "CHECKSUMMER V3"      PRG
32    "APFELMANN.PIC"       PRG
41    "0. PRG"              PRG
26 BLOCKS FREE.
READY.
```

Bild 3. Ein normales Diskettendirectory

```
0 "64'ER 11/85        " 64'ER
12    "CHECKSUMMER V3"      PRG
7     "MSE V1.0"            PRG
0     "----------------"    DEL
13    "HARDCOPY KOALA"      PRG
0     "----------------"    DEL
69    "LYRIC 3.0"           PRG
0     "----------------"    DEL
41    "0. PRG"              PRG
16    "1. PRG"              PRG
12    "2. PRG"              PRG
0     "  HYPRA PLATOS  "    PRG
0     "----------------"    DEL
69    "PROFIPRINT"          PRG
0     "----------------"    DEL
```

Bild 4. So kann Ihr Directory nach der Behandlung mit dem Directorysorter aussehen

```
4     "APFELROUTINEN"       PRG
58    "APFELMANN"           PRG
32    "APFELMANN.PIC"       PRG
26 BLOCKS FREE.
```

Bild 4. Schluß

Directory mit System

Nach beendetem Einlesen erscheint links vom obersten File ein schwarzer Pfeil, der sich mit den Funktionstasten F3 nach oben und mit F5 nach unten bewegen läßt. Mittels F1 kann nun ein mit diesem Pfeil gekennzeichnetes Feld nach rechts herausgeschoben und mit F3 und F5 verschoben werden. An gewünschter Stelle wird es mit F1 wieder eingesetzt. Somit ist ein beliebiges Vertauschen aller Files im Directory möglich.

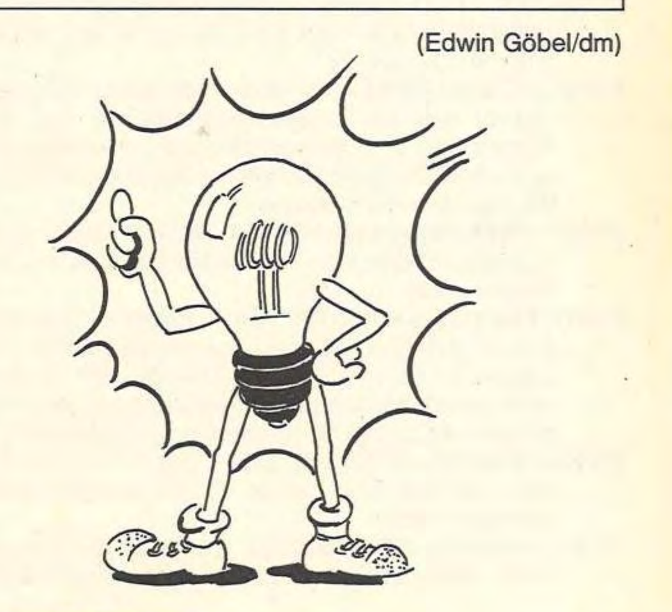
F8 schreibt das sortierte Directory wieder auf die Diskette zurück. Will man das geänderte Directory nicht gespeichert haben, so kann statt dessen mit F6 noch einmal das alte oder ein anderes eingelesen werden.

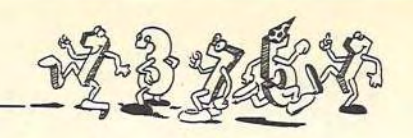
Zur optischen Abgrenzung mehrerer Files dient der Trennstrich, den man mit F2 erzeugen, mit F3 und F5 verschieben und schließlich mit F1 einfügen kann. Wem übrigens ein anderer Trennstrich besser gefällt, der kann in Zeile 260 für die Minuszeichen andere einsetzen, zum Beispiel SHIFT und +. SHIFT und C ist nicht zu empfehlen, da dieser Strich bei Groß-/Kleinschrift-Umschaltung ein großes C ergibt. Der Trennstrich belegt keinen Block auf der Diskette und ist zur besseren Unterscheidung mit DEL im Directory gekennzeichnet. Ein versehentlich mit F2 erzeugter Trennstrich kann, wenn er nach rechts gebracht wird, mit F4 wieder gelöscht werden. In gleicher Weise können auch Files aus dem Directory gelöscht werden. Dabei werden aber die von dem File belegten Blöcke nicht wieder freigegeben, so daß ein abschließendes VALIDATE bei der Diskette erforderlich ist, falls ein File gelöscht wurde.

Hier noch eine Zusammenstellung der Funktionstastenbelegung:

Taste		Funktion
F1	–	Eintrag markieren/einordnen
F2	–	Trennstrich erzeugen
F3	–	Gekennzeichnetes File aufwärts bewegen
F4	–	Gekennzeichnetes File löschen
F5	–	Gekennzeichnetes File abwärts bewegen
F6	–	Directory nochmal einlesen
F8	–	Geändertes Directory auf Diskette zurückschreiben

(Edwin Göbel/dm)





Die PEEK-, POKE- und SYS-Kiste

Das Betriebssystem bietet einige Kniffe, die sich erst durch spezielle POKEs richtig ausnutzen lassen. Auch andere spezielle und nützliche Routinen benötigen mindestens einen SYS-Aufruf, um arbeiten zu können. Lernen Sie hiermit die Tricks des C64 kennen.

Durch geschicktes POKEn in bestimmte Speicheradressen lassen sich verblüffende Effekte erzielen. So kann man etwa Programme gegen LISTen schützen, auf daß sie sich bei einem versuchten LIST-Befehl selbst zerstören. Dies kann recht hilfreich sein, wenn man sein eigenes Programm vor fremden Augen verbergen will. Aber lesen Sie selbst, welche Möglichkeiten sich noch bieten und lernen Sie dadurch Ihren C64 besser kennen.

POKE 1,53 – Das Basic und Kernel sind ausgeschaltet (die vollen 64 KByte RAM wären vorhanden – wenn aber das Betriebssystem nicht vorher vom ROM in das RAM kopiert wurde, stürzt der Computer ab, da er auf keine Befehle mehr zurückgreifen kann). Diese Konfiguration ist nur von Maschinensprache aus nutzbar.

POKE 1,54 – Hierbei ist das Basic ausgeschaltet. Diese Ebene ist ebenfalls nur unter Maschinensprache zu nutzen.

POKE 1,55 – Der normale Zustand. Der Interpreter und das Betriebssystem sind aktiv.

PRINT PEEK(10) – Ist der Wert = 0, so war die letzte Operation LOAD. VERIFY ergibt den WERT = 1.

PRINT PEEK(17) – Abfrage auf die letzte zugewiesene Variable. Bei dem Wert = 0 wurde die letzte Variable mittels INPUT geholt oder es fand noch keine Operation statt. Die Zuweisung mit GET ergibt den Wert = 64 und READ den Wert = 152.

POKE 19,64 – Beim nächsten INPUT-Befehl wird kein Fragezeichen mehr ausgegeben. Allerdings kann man nachher durch Drücken der RETURN-Taste nicht mehr in die nächste Zeile gelangen.

POKE 19,0 – Läßt die Ausgabe des Fragezeichens wieder zu.

POKE 24,0 – Nach Eingabe dieses POKEs auf einen FORMULA TOO COMPLEX-ERROR verhält sich der Computer wieder normal.

PRINT PEEK(43)+PEEK(44)*256 – Mit dieser Befehlszeile erfährt man die aktuelle Startadresse des Basic-Programms. Soll die Startadresse verändert werden, so muß in die Speicherzellen 43 und 44 ein anderer Wert geschrieben werden.

PRINT PEEK(45)+PEEK(46)*256 – Gleichlautend dazu erfährt man hiermit die aktuelle Endadresse des Basic-Programmes.

PRINT PEEK(55)+PEEK(56)*256 – Ergibt das derzeitige Ende des Basic-Bereiches. Durch Herabsetzen dieses Zeigers ist es möglich, den RAM-Bereich zu verkleinern, um etwa im oberen Bereich liegende Maschinenprogramme gegen Überschreiben zu schützen.

PRINT PEEK(57)+PEEK(58)*256 – Ergibt die Zeilennummer, bei der nach einer Programmunterbrechung gestoppt wurde.

PRINT PEEK(61)+PEEK(62)*256 – Dies ist der Zeiger auf CONT. Normalerweise ist nach einem anschließenden CLR das Programm nicht mehr fortzusetzen. Merkt man sich jedoch die Inhalte der beiden Speicherzellen noch vor dem CLR, so kann man anschließend an den Befehl die gemerkten Werte wieder hineinPOKEn und das Programm mit CONT wieder starten.

PRINT PEEK(63)+PEEK(64)*256 – So erhält man die Nummer der DATA-Zeile, aus der das zuletzt gelesene Datum geholt wurde. (Gut zum Finden von Fehlern in DATA-Zeilen geeignet.)

PEEK(69)/PEEK(70) – Name der zuletzt zugewiesenen Variablen. Bei normalen Fließkommavariablen ermittelt man den Wert mit PRINT CHR$(PEEK(69))+CHR$(PEEK(70)) aus.

Der Name von Integervariablen (zum Beispiel X%) ergibt sich durch PRINT CHR$(PEEK(69)-128)+CHR$(PEEK(70)-128).

Strings (zum Beispiel A$) erhält man durch PRINT CHR$(PEEK(69))+CHR$(PEEK(70)-128).

POKE 120,2 – Der C64 nimmt keinerlei Befehle mehr an.

POKE 147,X – Springt man die LOAD-Routine im Betriebssystem an, so entscheidet der Wert dieser Speicherstelle darüber, ob ein LOAD (X = 0) oder VERIFY (X = 4) durchgeführt wird.

PRINT PEEK(152) – Gibt die Anzahl der geöffneten Dateien bekannt.

PRINT PEEK(153) – Nummer des aktuellen Eingabegerätes. Hierbei gilt:

0	= Tastatur
1	= Datasette
2	= RS232-Port
3	= Bildschirm
8-11	= Floppy

PRINT PEEK(154) – Man erfährt hier die Nummer des aktuellen Ausgabegerätes.

0	= Tastatur
1	= Datasette
2	= RS232-Port
3	= Bildschirm
4,5	= Drucker
6	= Plotter
8-11	= Floppy

POKE 157,128 – Meldungen wie LOADING... oder SEARCHING... auch im Programmodus.

POKE 157,192 – Läßt alle Meldungen des Betriebssystems zu.

PRINT PEEK(182) – Dieser Befehl gibt die Anzahl der Zeichenlesefehler aus.

PRINT PEEK(183) – Länge des derzeitigen Filenamens.

PRINT PEEK(184) – Hiermit kann die laufende Filenummer abgerufen werden (OPEN...).

PRINT PEEK(185) – Analog hierzu läßt sich die derzeitige Sekundäradresse herausfinden.

PRINT PEEK(186) – ergibt die Gerätenummer des zuletzt angesprochenen Gerätes.

PRINT PEEK(197) – Derzeit gedrückte Taste

POKE 198,0 : WAIT 198,1 – Der Computer wartet auf einen Tastendruck.

POKE 199,1 – Umschaltung in den Revers-Modus. Das Rücksetzen in die Normalausgabe erfolgt mit POKE 199,0.

PRINT PEEK(200) – Gibt an, wieviele Zeichen die zuletzt eingegebene Zeile hatte.

PRINT PEEK(201) – Gibt die Zeile der aktuellen Cursorposition an.

PRINT PEEK(202) – So erhält man die Spalte der aktuellen Cursorposition.

PRINT PEEK(203) – Gibt an, welche Taste gedrückt wurde, und zwar im Charaktermodus (siehe Tabelle im C64-Handbuch), wobei 64 bedeutet, daß keine Taste gedrückt wurde.

POKE 204,0 – Der Cursor bleibt auch bei GET an.
POKE 204,1:POKE 207,0 – Dieses normalisiert den oberen Zustand wieder.
POKE 212,0 – Flag für Gänsefüßchenmodus
PRINT PEEK(215) – Hiermit läßt sich das zuletzt eingegebene Zeichen im ASCII-Code auslesen.
POKE 646,X – setzt die aktuelle Cursorfarbe (Wert für X von 0 bis 15).
POKE 647,X – Über die X-Werte von 0 bis 15 läßt sich die jeweilige Farbe unter dem Cursor ändern.
POKE 648,X – Beginn Bildschirmspeicher
POKE 649,0 – Sperren von Tastatureingaben.
POKE 649,10 – Rücksetzung auf den Normalzustand.
POKE 650,255 – Ab jetzt haben alle Tasten REPEAT-Funktion, das heißt Dauerfunktion.
POKE 650,0 – Rücksetzung in den Normalzustand.
POKE 650,127 – Dauerfunktion der Cursortasten ausschalten.
POKE 650,64 – Dauerfunktion für alle Tasten aus.
PRINT PEEK(653) – Abfrage auf SHIFT (Bit 0)-, COMMODORE (Bit 1)- und CONTROL (Bit 2)-Taste.
POKE 657,128 – Dieser Befehl macht die Umschaltung von der Tastatur her auf Groß- und Kleinschreibung unmöglich.
POKE 768,145 – Fehlermeldungen abschalten
POKE 768,139 – normal
POKE 768,143 – Reset nach Programmende
POKE 770,X – Durch EinPOKEn eines beliebigen Wertes erfolgt die READY-Meldung unendlich oft. Nur noch Ausschalten hilft.
POKE 774,226:POKE 775,252 – Wird nun der LIST-Befehl angewandt, so erfolgt ein Reset des Computers.
POKE 775,200 – So kann man einen kleinen Listschutz aktivieren, der sich mit POKE 775,167 wieder desaktivieren läßt.
POKE 776,1 – Dieser Befehl zerstört das im Speicher befindliche Basic-Programm.
POKE 777,1 – Ab jetzt wird kein Befehl mehr ausgeführt. Der Cursor steht in der linken oberen Ecke.
Stelle **781/782** – Startadresse, ab der ein Programm geladen wird. Durch entsprechende Werte kann ein Programm in einen anderen Speicherbereich geladen werden. (Siehe SYS 65493)
POKE 788,52 – Dieser Befehl hebt die Wirkung der STOP-Taste auf.
POKE 788,49 – STOP-Taste wieder einschalten
POKE 788,62 – Cursor blinkt schneller
Stelle **788/789** – IRQ-Vektor. Das Betriebssystem springt ständig in diese Routine. Durch Ändern des Inhalts können eigene, »interruptgesteuerte« Maschinenroutinen ständig neben dem Hauptprogramm laufen.
POKE 792,226:POKE 793,252 – wirkt wie ein RESET beim Drücken der Restore-Taste.
POKE 801,0:POKE 802,0:POKE 818,165 – Nach diesen drei POKE-Befehlen kann ein im Speicher befindliches Programm weder auf Kassette noch auf Diskette gespeichert werden.
POKE 808,225 – Hebt die Tastenkombination RUN/STOP und RESTORE auf. Ein Programm läßt sich nicht mehr unterbrechen.
POKE 808,237 – RUN/STOP und RESTORE werden wieder aktiviert.
POKE 813,2 – Nach der Eingabe dieses Befehles ist ein Basic-Programm nicht mehr zu ändern.
POKE 818,226:POKE 819,252 – Wird der Befehl SAVE eingegeben, so löst der Computer einen Reset aus.
POKE 53270,X – Durch verschiedene X-Werte kann ein horizontales Soft-Scrolling erreicht werden. X = 1: Scrollen nach rechts. X = 0: Scrollen nach links.
POKE 53280,X – Ein Ändern des Wertes X im Bereich von 0 bis 15 setzt eine andere Rahmenfarbe des Bildschirms.
POKE 53281,X – Gleichlautend zum oberen Befehl wird die Hintergrundfarbe geändert.
Stelle **53260** – Joystick Port 2:
WAIT 56320,16,16 wartet auf Druck des Feuerknopfes.
WAIT 56320,4,4 – Warten auf Linksbewegung des Joysticks.
WAIT 56320,1,1 – Warten auf Joystick nach oben.
WAIT 56320,2,2 – Warten auf Joystick nach unten.
WAIT 56320,8,8 – Warten auf Rechtsbewegung des Joysticks.
Stelle **56321** – Analog zu Speicherstelle 56320, nur Port 1.
POKE 56325,5 – Der Cursor wird rasend schnell.
POKE 56325,49 – Normalisierung der Cursor-Geschwindigkeit.
POKE 56325,255 – Der Cursor bewegt sich sehr langsam.
PRINT PEEK(56576) – Abfrage der Pins am Userport. Nähere Informationen darüber im Artikel »Die Ports des C 64« in diesem Heft.
POKE 56576,X – Setzen der Ausgaben auf dem Userport. Nähere Informationen finden Sie ebenfalls im oben genannten Artikel.
POKE 56578,X – Datenrichtungsregister Userport. Hiermit wird entschieden, ob die Pins des Userports auf Ein- oder Ausgabeoperationen arbeiten. Informationen darüber im oben erwähnten Artikel.
SYS 42039 – Ausgabe von Fehlermeldungen. Dabei muß in 781 die Fehlernummer gePOKEd werden.
SYS 43121 – Entspricht dem Befehl RUN 0
SYS 45499 – Mittels dieses Befehls wird die interne Uhr auf Null gesetzt.
POKE 214,(Zeile):POKE 211,(Spalte):SYS 58640 – So läßt sich der Cursor ohne umständliche PRINT-Anweisungen auf jede Stelle des Bildschirms positionieren.
SYS 63123 – Dieser Befehl kann eine große Hilfe sein. Angenommen, ein Programm will ein anderes von der Diskette nachladen, findet es aber nicht. Das Programm steigt in diesem Fall mit der Fehlermeldung FILE NOT FOUND aus und die rote LED an der Floppy beginnt zu blinken. Durch diesen SYS können Sie nun feststellen, wie das File heißt, das gesucht wurde.
SYS 64738 – Hiermit wird der Computer in seinen Einschaltzustand versetzt. Das bedeutet: Programme werden gelöscht, Vektoren und Register zurückgesetzt.
SYS 64763 – Führt das gleiche wie SYS 64738 aus bis auf den Vektor für den IRQ, der auf seinem Wert belassen wird.
SYS 65409 – Setzt den Video-Chip wieder in den Ursprungszustand zurück.
SYS 65511 – Dieser Befehl schließt alle zu diesem Zeitpunkt offenen Files. So erspart man sich das lästige Eintippen von CLOSE 1:CLOSE 2:CLOSE 3:... Dabei sollte aber beachtet werden, daß so nur die Kanäle geschlossen werden, aber keine Dateien auf Diskette.
SYS 65493 – LOAD-Routine des Betriebssystems. Mit folgender kleiner Routine kann man Unterprogramme nachladen, ohne irgendwelche Basic-Pointer (zum Beispiel die Zeiger auf die Endadresse, 45 und 46) zu verändern:

```
POKE 186,1:POKE 780,0:POKE 781,0:
POKE 782,96: POKE 183,0:SYS 65493
```

Erklärung: 186,1 = Geräteadresse für Datasette
781 und 782 geben die Startadresse an, ab der das Programm nachgeladen werden soll.
183,0 = kein Programmname
SYS 65493 = LOAD-Routine

(dm)



Mein Computer versteht mich nicht!

Jeder Programmierer, ob Anfänger oder Profi, macht beim Programmeschreiben Fehler. Der Anfänger jedoch kennt meist den genauen Grund, und wie er ihn beheben kann, nicht. Dem wollen wir mit einer ausführlichen Fehlertabelle und vielen Tips zur Suche der Fehlerursache abhelfen.

Obwohl dieser Artikel eigentlich mehr als Nachschlagewerk für den Notfall gedacht ist, schadet es nicht, wenn Sie ihn einmal komplett durchlesen. Sie haben so ein »Grundwissen«, auf das Sie bei Bedarf zurückgreifen können. Auch können Sie dann auf Zusammenhänge zwischen den einzelnen Fehlern schließen. Doch kommen wir nun zum Praxisteil:

Sie haben sich einen neuen C 64 besorgt, packen ihn voller Erwartung zu Hause aus, schließen ihn an und los geht's mit dem Programmieren. Aber o weh!, der Computer macht nicht das, was man von ihm verlangt. Es erscheinen Fehlermeldungen auf dem Bildschirm (erkennbar am vorangestellten »?« und daran, daß sich der Computer unerwartet mit »READY.« zurückmeldet). Was tun? Spätestens jetzt wird die Bedienungsanleitung zum C 64 zur Hand genommen, um dem Fehler auf die Schliche zu kommen. Aufmerksam liest man das entsprechende Kapitel, doch was versteht man? Nichts!! Lauter Fachausdrücke, von denen man noch nie etwas gehört hat, helfen auch nicht weiter. Genau an diesem Punkt soll dieser Artikel ansetzen, um Ihnen bei der Fehlersuche und deren Beseitigung unterstützend unter die Arme zu greifen. Jeder mögliche Fehler wird ausführlich behandelt. Dabei ist jedem Fehler ein Absatz gewidmet, der wie folgt aufgebaut ist:

- Fehlermeldung, also das, was auf dem Bildschirm erscheint und kurze Erklärung des Fehlers
- Klärung der Fachausdrücke
- Wie entsteht der Fehler, mit Beispielen?
- Wie geht man vor, um die Fehlerursache zu suchen?
- Wie wird der Fehler beseitigt?

Hinweis: Bevor Sie ein neues Beispiel eintippen, sollten Sie erst das alte mit »NEW« löschen.

Fehler im Betriebssystem:

Wenn Sie mit einer RS232-Schnittstelle arbeiten, so ist folgendes zu beachten: Nach einem `OPEN 1,2,...` also »Datenkanal zur RS232-Schnittstelle öffnen«, werden vom Computer sämtliche Variablen eines Basic-Programms gelöscht! Daher sollte dieser OPEN-Befehl unbedingt als erster Befehl in einem solchen Programm stehen.

?BAD SUBSCRIPT:

Der Fehler tritt auf, wenn versucht wird, eine Feldvariable anzusprechen, die nicht dimensioniert wurde.

Eine Feldvariable ist eine normale Variable wie »A« oder »TEST« mit zusätzlichem Index wie »A(5)« oder »TEST(X)«. Der Index in Klammern ist als Zeiger zu verstehen, der auf ein Element des Feldes mit dem Namen »A« oder »Test« zeigt. Felder, die mehr als elf Elemente enthalten, müssen mit dem Basic-Befehl DIM definiert werden. Felder, die weniger als elf Elemente enthalten, müssen nicht definiert werden. Zum Beispiel legt der Befehl DIM A(20) ein Feld mit insgesamt 21 Elementen fest (21 Elemente, weil die Feldvariable A(0) auch in der Dimensionierung enthalten ist). Wenn nach dieser Felddefinition versucht wird, der Feldvariablen »A(21)« einen Wert zuzuweisen »A(21)=123«, erscheint die Fehlermeldung »BAD SUBSCRIPT« auf dem Bildschirm. Die gleiche Fehlermeldung erhält man, wenn der Feldvariablen »A(11)« ein Wert

zugewiesen wird, ohne zuvor ein entsprechend großes Feld zu dimensionieren.

Tritt ein solcher Fehler in einem komplexeren Programm auf, ist zuerst zu überprüfen, wie groß der Index der Feldvariablen ist. Sollte er größer als 10 sein, ist die dazugehörige DIM-Anweisung zu suchen. Wenn eine solche nicht existiert, erweitern Sie eine der ersten Zeilen eben um diese DIM-Anweisung.

BREAK:

Bei dieser Meldung handelt es sich nicht um eine Fehlermeldung im herkömmlichen Sinn. Vielmehr macht der Computer darauf aufmerksam, daß das Programm oder ein Speicher- oder Ladevorgang absichtlich unterbrochen wurde. Diese Meldung existiert in zwei verschiedenen Versionen.

?BREAK ERROR zeigt an, daß beim Laden oder Speichern eines Programms die RUN/STOP-Taste gedrückt wurde.

?BREAK IN XXXX weist darauf hin, daß während eines Programmlaufes die RUN/STOP-Taste gedrückt wurde, oder daß sich in der angezeigten Zeile der Basic-Befehl »STOP« oder »BREAK« befindet. In einem solchen Fall kann das Programm mit dem Basic-Befehl »CONT« für »continue« fortgesetzt werden, vorausgesetzt es erscheint nicht die Fehlermeldung

?CAN'T CONTINUE: In einem solchen Fall ist einer der folgenden Punkte eingetreten:

1) Das Programm wurde nicht, wie oben beschrieben, absichtlich unterbrochen, sondern der Computer selbst hat es angehalten, weil ein »SYNTAX ERROR« auftrat (der Basic-Übersetzer konnte einen Befehl im Programm nicht übersetzen/verstehen).

2) Im Direktmodus wurden mit dem Basic-Befehl »CLR« die Variablen gelöscht; das heißt Sie haben, nachdem das Programm unterbrochen wurde, folgendes eingegeben:

```
CLR <RETURN>
```


3) Am Programm wurde nach einer Unterbrechung etwas geändert.

4) Im Direktmodus trat nach einer Unterbrechung ein Fehler auf.

Beispiel: Sie Starten ein beliebiges Basic-Programm, halten es mit der RUN/STOP-Taste an und geben ein

```
A <RETURN>
```

Der Computer meldet einen »SYNTAX ERROR«. Klar, den Befehl »A« gibt es nicht. Nun versuchen Sie das Programm mit »CONT« fortzusetzen. Der Computer gibt die Fehlermeldung »CAN'T CONTINUE« aus.

5) Wenn Sie den Befehl »CONT« eingeben, bevor ein Basic-Programm gestartet wurde, reagiert der Computer ebenfalls mit »CAN'T CONTINUE«.

?DEVICE NOT PRESENT:

Es wurde versucht, ein peripheres Gerät wie Drucker, Diskettenlaufwerk oder ähnliches anzusprechen, das entweder nicht eingeschaltet oder überhaupt nicht vorhanden ist oder nicht reagiert. Dieser Fehler tritt gewöhnlich nicht beim eigentlichen »OPEN«-Befehl auf, wie das folgende Beispiel zeigt, sondern dann, wenn versucht wird, das Gerät mit den Befehlen »GET#, INPUT#, PRINT#« anzusprechen. Beispiel:

```
OPEN 7,7 <RETURN>
PRINT#7,"ABCDEF" <RETURN>
```

Meldet der Computer einen »DEVICE NOT PRESENT«-Error, sollten Sie zunächst überprüfen, ob alle erforderlichen Geräte (Drucker, Diskettenlaufwerk und so weiter) eingeschaltet und mit dem Computer verbunden sind. Ist das der Fall, haben Sie mit Sicherheit eine falsche Geräteadresse (zweite Zahl hinter dem OPEN-Befehl) gewählt. Um den für die Fehlermeldung verantwortlichen OPEN-Befehl zu finden, lassen Sie sich die in der Fehlermeldung angegebenen, Zeile LISTen. Suchen Sie in dieser Zeile einen Basic-Befehl, dem unmittelbar das Nummernzeichen (#) und eine Zahl (Filenummer) folgt. Sind mehrere solcher Befehle vorhanden, ist die Zeile aufzuteilen und zwar so, daß jede Zeile nur einen Befehl mit Nummernzeichen enthält. Beispiel:

```
10 PRINT#4,CHR$(32):GET#8,A$
```

Diese Zeile ist in die folgenden zwei Zeilen aufzuteilen:

```
10 PRINT#4,CHR$(32)
11 GET#8,A$
```

Starten Sie nun das Programm erneut mit RUN und lassen sich wieder die in der Fehlermeldung angegebene Zeile LISTen. Im nächsten Schritt ist der OPEN-Befehl im Programm zu suchen, der die gleiche Filenummer (erste Zahl hinter dem OPEN-Befehl) hat, wie der Befehl (Zahl hinter dem Nummernzeichen), bei dem der Fehler aufgetreten ist. Die zweite Zahl hinter dem fehlerträchtigen OPEN-Befehl ist die falsche Geräteadresse. Damit Sie dort die richtige Geräteadresse einsetzen können, folgt eine Tabelle aller möglichen Geräteadressen:

1 Datasette
2 RS232
3 Bildschirm
4 Drucker
5 Drucker, falls im Drucker die Geräteadresse auf 5 umgestellt wurde
6 Plotter 1520
7 frei für externe Geräte
8 bis 11 Diskettenlaufwerke
12 bis 15 frei für externe Geräte

?DIVISION BY ZERO:

Dieser Fehlermeldung erscheint nur dann, wenn durch Null dividiert wurde. Aber Vorsicht! Es ist nicht immer auf Anhieb ersichtlich, daß der Nenner tatsächlich Null ist. Ist zum Beispiel bei der Funktion »A=B/(C*D*E*F)« eine der Variablen »C, D, E, F« Null, so ist natürlich der gesamte Nenner Null. Dieses Beispiel ist vielleicht trivial, aber was machen Sie, wenn bei der Funktion »A=TAN (PI/2)« ein »DIVISION BY ZERO«-Error erscheint? Um das zu verstehen, muß man wissen, wie der Computer die TAN-Funktion behandelt. Er berechnet nämlich nicht direkt den Tangens sondern ermittelt ihn mit Hilfe der SIN- und COS-Funktion (tan(x) = sin(x)/cos(x)). Wenn für X $\pi/2$ eingesetzt wird ist der Nenner »COS ($\pi/2$)« Null, und es erscheint die Fehlermeldung.

Um den »DIVISION BY ZERO«-Error zu beheben, ist jede Variable im Nenner, falls vorhanden, mit dem Basic-Befehl »PRINT«, gefolgt von der Variablen, zu überprüfen. Ist die Variable gefunden, die den Fehler verursacht hat, müssen Sie dafür sorgen, daß sie niemals den Wert Null annehmen kann. Dies läßt sich durch eine Abfrage vor der Funktion realisieren. Beispiel:

```
IF C*D*E*F <>0 THEN A=B/(C*D*E*F)
```

?EXTRA IGNORED:

Zeigt an, daß hinter dem Basic-Befehl INPUT weniger Variablen stehen, als eingegeben wurden. Beispiel:

```
10 INPUT A$,B$ <RETURN>
```

Nach der Eingabe wird dieser Einzeiler mit RUN <RETURN> gestartet. Es erscheint ein Fragezeichen (?) auf dem Bildschirm. Geben Sie nun ein:

```
TEST1,TEST2,TEST3 <RETURN>
```

Der Computer meldet einen »EXTRA IGNORED«-Error, denn »TEST1« wird in die Variable »A$« und »TEST2« in die Variable »B$« eingelesen. Für »TEST3« existiert keine Variable und genau dies meldet der Computer. Häufig entsteht eine solche Fehlermeldung unbeabsichtigt bei Zeichenketteneingaben, die Kommata enthalten. Man sollte daher darauf achten, wenn mit dem INPUT-Befehl gearbeitet wird, daß eine Zeichenkette, die in eine Variable eingelesen werden soll, kein Komma enthält. Durch das Komma wird nämlich dem Computer mitgeteilt, daß alle folgenden Zeichen bis zum nächsten Komma in die nächste Variable eingelesen werden sollen. Eine Zeichenkette darf neben dem Komma auch keine

Doppelpunkte enthalten; denn alle Zeichen, die hinter dem ersten Doppelpunkt stehen, werden vom Computer verschluckt. Der gleiche Fehler tritt auch bei dem Befehl »INPUT#« auf, jedoch wird in diesem Fall keine Fehlermeldung ausgegeben. Übrigens noch ein Tip: Bei älteren C64 darf der Cursor nicht nach oben oder unten verschoben werden, wenn der INPUT-Befehl eine Eingabe verlangt.

?FILE DATA ERROR:

Diese Fehlermeldung erscheint, wenn im Programm versucht wird, Zeichenketten vom Diskettenlaufwerk, von der Datasette oder der Tastatur in eine numerische Variable einzulesen. Beispiel:

```
10 OPEN1,0:REM TASTATUR ALS EINGABEGERÄT ÖFFNEN
20 INPUT#1,A:REM AUF EINGABE WARTEN
30 CLOSE1:REM EINGABEGERÄT »TASTATUR« SCHLIESSEN
```

Zeile 10 öffnet einen Kanal mit der Filenummer 1 und setzt die Tastatur (Geräteadresse 0) als Eingabegerät.

In Zeile 20 wird auf die Eingabe gewartet. Es können beliebige Zahlen (keine Buchstaben) eingegeben werden. Nach dem Drücken der RETURN-Taste wird in Zeile 30 der Kanal mit der Filenummer 1 geschlossen.

Sind die drei oben stehenden Zeilen abgetippt, läßt sich das Programm mit RUN starten. Im Gegensatz zum normalen INPUT-Befehl ohne dem Nummernzeichen wird kein Fragezeichen ausgegeben. Geben Sie nun ein:

```
123 <RETURN>
456789 <RETURN>
```

und so weiter

Es erscheint, wie erwartet, keine Fehlermeldung. Nun versuchen Sie es mal mit »1A3«. Der Computer meldet einen »FILE DATA«-Error. Die erste Zahl »1« kann der numerischen Variablen »A« zugewiesen werden. Das zweite Zeichen jedoch ist keine Zahl, sondern ein alphanumerisches Zeichen beziehungsweise ein Buchstabe und läßt sich nicht einer numerischen Variablen zuordnen.

Der angezeigte Fehler kann sehr schnell gefunden werden, da die Zeile, in der der Fehler auftrat, mit der Fehlermeldung zusammen ausgegeben wird.

?FILE NOT FOUND:

Es wurde versucht, ein Programm- oder Daten-File von der Diskette zu laden, dessen Name nicht auf der momentan eingelegten Diskette existiert. Um diese Fehlermeldung kennenzulernen, tippen Sie einfach ein »LOAD"ALR",8«. Natürlich muß dazu ein Diskettenlaufwerk vorhanden, eingeschaltet und mit dem Computer verbunden sein. Erscheint eine solche Fehlermeldung innerhalb eines Programms, wird zusammen mit der Fehlermeldung die Zeile ausgegeben, in der der Fehler auftrat. Sollte in der fehlerhaften Zeile statt eines Namens hinter dem LOAD-Befehl nur eine alphanumerische Variable stehen, zum Beispiel »...:LOAD A$,8:...«, dann läßt sich dieser Name mit »PRINT A$« auf den Bildschirm bringen. Wenn sich aus irgendeinem Grund der Name nicht mehr feststellen läßt, kann er mit dem Befehl »SYS63123« noch einmal angezeigt werden. Übrigens: Die Meldung »FILE NOT FOUND« erscheint nur, wenn mit dem Diskettenlaufwerk gearbeitet wird. Bei der Datasette wird entweder nichts oder »DEVICE NOT PRESENT« ausgegeben.

?FILE NOT OPEN:

Es wurde versucht, mit den Befehlen »PRINT#, GET#, INPUT#« Zeichen an ein externes Gerät zu übermitteln oder von einem solchen zu lesen, ohne zuvor das File mit dem OPEN-Befehl geöffnet zu haben. Dabei ist es durchaus denkbar, daß eine falsche Filenummer (erste Zahl hinter dem OPEN-Befehl) hinter einem der oben stehenden Befehle oder hinter dem OPEN-Befehl eingesetzt wurde. Beispiel:

```
10 OPEN 5,4: REM DRUCKERKANAL ÖFFNEN
20 PRINT#4,"ABC":REM UND DIE ZEICHEN »ABC« DRUCKEN
30 CLOSE 4: REM DRUCKERKANAL SCHLIESSEN
```

Wird dieses Programm eingegeben und gestartet, so erhält man einen »FILE NOT OPEN«-Error, denn die Zeile öffnet einen Kanal mit der logischen Filenummer »5«. Die Zeilen 20 und 30 beziehen sich aber auf die logische Filenummer »4«.

Bei längeren ineinander verschachtelten Programmschleifen ist es häufig sehr schwierig, einen solchen Fehler zu beseitigen. Daher geht man einen anderen Weg. Man öffnet direkt vor dem PRINT#-Befehl in der gleichen Zeile einen Kanal mit der entsprechenden Filenummer (im Beispiel wird die Zeile 20 so ergänzt: 20 OPEN4,4:PR...). Jetzt muß der Kanal natürlich innerhalb des Programms wieder geschlossen werden. Wenn Sie unsicher sind, daß das tatsächlich gemacht wird, ist unmittelbar hinter dem PRINT#-Befehl der Kanal zu schließen (im Beispiel muß wieder die Zeile 20 ergänzt werden: 20 ..:.."ABC":CLOSE 4). Tritt der gleiche Fehler nach dem Start des Programms in einer anderen Zeile auf, in der ebenfalls die gleiche Filenummer benutzt wurde, so muß der zuletzt eingefügte CLOSE-Befehl gelöscht und hinter dem Befehl eingesetzt werden, der den neuen Fehler verursacht hat. So können Sie sich durch das gesamte Programm »hangeln«, bis es fehlerfrei arbeitet.

?FILE OPEN:

Diese Fehlermeldung ist genau das Gegenstück zu der Fehlermeldung »FILE NOT OPEN«. Erscheint der »FILE OPEN«-Error auf dem Bildschirm, wurde ein Kanal mit der gleichen Filenummer zweimal geöffnet. Das heißt es fehlt ein CLOSE-Befehl, der den zuerst geöffneten Kanal wieder schließt. Beispiel:

```
10 OPEN 4,4
20 OPEN 4,4
```

Der einfachste Weg, diesen Fehler zu beseitigen, ist es, vor dem OPEN-Befehl, in der Zeile, die hinter der Fehlermeldung angegeben ist, den entsprechenden Kanal mit dem CLOSE-Befehl zu schließen (im Beispiel: 20 CLOSE 4:OPEN 4,4). Sie werden sich jetzt zu Recht fragen, warum wird nicht einfach die Zeile 20 gelöscht? Die Antwort darauf ist ganz einfach. Hinter dem ersten OPEN-Befehl mit der Filenummer 4 könnte ja auch eine andere Geräteadresse stehen, durch die ein anderes Gerät angesprochen wird (zum Beispiel OPEN 4, 8 für Diskettenlaufwerk). Würde jetzt der zweite OPEN-Befehl (im Beispiel Zeile 20) gelöscht werden, so würde alles, was zum Drucker (Geräteadresse 4) gesendet werden soll, zum Diskettenlaufwerk geschickt. Daher seien Sie äußerst vorsichtig mit dem Löschen von Befehlen innerhalb eines Programms.

?FORMULA TOO COMPLEX:

Es ist fast unmöglich, einen solchen Fehler in ein Basic-Programm einzubauen. Trotzdem existiert die Fehlermeldung. Durch sie meldet der Computer, daß in einer Zeichenkettenaddition zu viele Klammerausdrücke stehen. Beispiel:

```
PRINT "A"+("B"+"B1"+("C"+"D"))
```

Die Anzahl der Zeichen innerhalb der einzelnen Zeichenketten ist völlig uninteressant. Wichtig ist, daß zweimal die Zeichenkombination »+(« vorkommt. Denn der Computer versucht sich alles vor den jeweiligen Klammerausdrücken zu merken. Im Beispiel merkt sich der C64 das Zeichen »A«, berechnet »"B" + "B1" = "BB1"« und versucht sich auch diese Zeichen zu merken, um im nächsten Schritt »"C" + "D"« auszurechnen. Soweit kommt er aber nicht, weil er sich maximal eine Zeichenkette merken kann (im Beispiel »A«) und gibt daher die Fehlermeldung »FORMULA TOO COMPLEX« aus.

Anmerkung: Verschachtelungen innerhalb einer Zeichenkettenaddition sind völlig überflüssig. Sie erhalten nur dann einen Sinn, wenn nicht nur addiert, sondern auch multipliziert oder dividiert wird und genau dies ist mit Zeichenketten nicht möglich.

Die oben stehende Zeile könnte daher auch so aussehen:

```
PRINT "A"+"B"+"B1"+"C"+"D"
```

oder einfach

```
PRINT "A";"B";"B1";"C";"D"
```

oder noch einfacher

```
PRINT "A" "B" "B1" "C" "D"
```

Das Ergebnis ist immer das Gleiche. Deshalb ist bei Zeichenkettenoperationen das Pluszeichen und vor allen Dingen das Arbeiten mit Klammern zu vermeiden.

Sollte trotzdem mal ein »FORMULA TOO COMPLEX«-Error erscheinen, ist ein fataler Fehler gemacht worden, der sich nur beseitigen läßt, indem man den Computer kurzzeitig ausschaltet.

?ILLEGAL DEVICE NUMBER:

Diese Fehlermeldung bedeutet, daß ein Befehl zu einem unzulässigen Gerät geschickt wurde. Mit anderen Worten: Die im Befehl angegebene Geräteadresse ist falsch. Zum Beispiel erzeugt »LOAD"TEST",0« einen »ILLEGAL DEVICE NUMBER«-Error, denn es ist unmöglich, ein Programm von dem Gerät mit der Geräteadresse »0« (gleich Tastatur) zu laden.

Erscheint diese Fehlermeldung auf dem Bildschirm, ist die Beseitigung des Fehlers recht einfach. Lassen Sie sich die Zeile, die hinter der Fehlermeldung angegeben ist, LISTen und schauen sich alle LOAD- und SAVE-Befehle an. Ist eine der Geräteadressen hinter diesem Befehl »0, 2 oder 3«, dann haben Sie den Fehler gefunden. Sollte nicht bekannt sein, welche Gerätenummer welchem Gerät zugeordnet ist, dann schauen Sie in der Gerätenummern-Tabelle nach, die bei der Beschreibung der Fehlermeldung »DEVICE NOT PRESENT« aufgeführt ist.

ILLEGAL DIREKT:

Wenn versucht wird, den Basic-Befehl »GET« oder »INPUT« im Direktmodus einzugeben, dann meldet sich der Computer mit dieser Fehlermeldung. Der Direktmodus unterscheidet sich vom Programmodus dadurch, daß im Direktmodus jeder eingegebene Befehl ausgeführt wird, sobald man die RETURN-Taste drückt. Den Programmodus erkennt der C64 an einer Zahl (Zeilennummer) am Anfang einer Zeile. Wird hinter einer Zeile, die mit einer Zeilennummer beginnt, die RETURN-Taste gedrückt, dann wird die Zeile nicht ausgeführt, sondern in den Speicher des C64 geschrieben. Beispiel:

```
INPUT A$ <RETURN>
```

führt zur Fehlermeldung »ILLEGAL DIREKT«

Wird vor den INPUT-Befehl eine Zahl geschrieben »10 INPUT A$ <RETURN>«, dann akzeptiert der Computer diese Zeile und speichert sie. Wenn der INPUT-Befehl jetzt ausgeführt werden soll, ist RUN <RETURN> im Direktmodus einzugeben.

Da der »ILLEGAL DIREKT«-Error niemals in einem Programm auftauchen kann, erübrigt sich die Behebung dieses Fehlers.

ILLEGAL QUANTITY:

Diese Fehlermeldung zeigt an, daß die Berechnung einer Funktion eine Zahl ergab, die der Computer in dieser Form nicht verarbeiten kann. Arbeitet man mit Integer-Variablen (zum Beispiel A%), ist darauf zu achten, daß den Variablen niemals Zahlen zugewiesen werden, deren Wert 32000 überschreitet. Wird versucht, mit dem Basic-Befehl POKE eine Zahl zu speichern, deren Wert größer ist als 255 oder negativ, dann erscheint auch ein »ILLEGAL QUANTITY«-Error. Die gleiche Fehlermeldung gibt der C64 auch dann aus, wenn beim OPEN-Befehl eine logische Filenummer (erste Zahl hinter dem OPEN-Befehl) größer als 255 gewählt wurde, oder wenn aus einer negativen Zahl die Quadratwurzel gezogen werden soll. Alle folgenden Beispiele erzeugen einen »ILLEGAL QUANTITY«-Error:

```
A%=33000:REM DIE INTEGERVARIABLE A% IST GRÖSSER
32000
POKE 2,300:REM ES SOLL EINE ZAHL GESPEICHERT
WERDEN, DIE GRÖSSER IST ALS 255
POKE 2,-3:REM NEGATIVE WERTE LASSEN SICH NICHT
SPEICHERN
OPEN 300,4:REM FILENUMMER GRÖSSER 255
PRINT SQR(-1):REM AUS NEGATIVEN ZAHLEN KANN NICHT
DIE QUADRATWURZEL GEZOGEN WERDEN
```

Einen solchen Fehler im Programm aufzuspüren ist nicht einfach, denn in den meisten Fällen enthalten Variable die fehlerträchtigen Zahlen. Um einen solchen Fehler zu beseitigen, lassen Sie sich zunächst die Zeile LISTen, die den Fehler enthält. Im Anschluß daran ist jede in dieser Zeile vorkommende Variable mit dem Befehl PRINT auf den Bildschirm zu schreiben. Wenn bei diesem Versuch die Fehlermeldung angezeigt wird, ist zumindest die fehlerhafte Variable gefunden. Jetzt muß gegebenenfalls Schritt für Schritt der Programmablauf verfolgt werden, um die Stelle zu finden, an der der Variablen die falsche Zahl zugewiesen wurde.

I/O ERROR 1 bis 29:

Diese Fehlermeldung existiert nach dem Einschalten des Computers nicht. Nur dann, wenn in die Speicherzelle 157 die Zahl 64 oder 192 geschrieben wird (zum Beispiel POKE 157,64), gibt der Computer im Falle eines Fehlers die Fehlermeldung »I/O ERROR« aus. Dabei entspricht eine Zahl zwischen 1 und 29 hinter der Fehlermeldung den normal angezeigten Fehlern wie zum Beispiel »ILLEGAL QUANTITY«. Beispiel:

```
POKE157,192:LOAD"ABC",9
```

Existiert kein Diskettenlaufwerk mit der Gerätenummer 9, so führt das Beispiel zu der Fehlermeldung:

```
SEARCHING FOR ABC
I/O ERROR #5
?DEVICE NOT PRESENT
```

?LOAD ERROR:

Ein Programm oder Daten-File konnte nicht fehlerfrei von der Datasette oder von dem Diskettenlaufwerk geladen werden. Meldet der Computer diesen Fehler, wenn ein Programm abgearbeitet wird, so liegt der Fehler nicht am Programm, sondern an dem Gerät, von dem es geladen wurde. Versuchen Sie daher das gleiche Programm noch einmal zu laden. Scheitert auch dieser Versuch, so ist das Programm verloren. Meistens kommt ein »LOAD ERROR« dann vor, wenn entweder die Diskette oder das Band der Kassette mechanisch beschädigt ist.

?MISSING FILE NAME:

Weist darauf hin, daß beim LOAD- oder SAVE-Befehl kein Name angegeben worden ist, denn die Angabe eines Namens ist bei Operationen mit dem Diskettenlaufwerk unbedingt erforderlich. »LOAD "",8 <RETURN>« führt folglich zur Fehlermeldung »MISSING FILE NAME«.

Tritt der Fehler innerhalb eines Programms auf, lassen Sie sich die Zeile, die hinter der Fehlermeldung angegeben ist, LISTen. Folgt dem entsprechenden LOAD- oder SAVE-Befehl ein Leerzeichen »" "«, dann ist zwischen den Gänsefüßen ein beliebiger Name einzusetzen. Schwieriger wird es, wenn der Befehl so aussieht:

```
...:load a$,8:...
```

Schauen Sie sich in einem solchen Fall den Inhalt der Stringvariablen (Zeichenkettenvariable) mit dem Befehl

```
PRINT A$ <RETURN>
```

an. Sie wird kein einziges Zeichen enthalten. Folglich muß unmittelbar vor dem entsprechenden Befehl der Stringvariablen A$ ein Name zugewiesen werden. Die Zeile (Beispiel) müßte dann so aussehen:

```
...:a$="zeichen":load a$,8:...
```

Häufig ist aber erwünscht, daß sich mit der gleichen Zeile unterschiedliche Programme laden lassen. In einem solchen Fall muß, bevor der LOAD- oder SAVE-Befehl ausgeführt wird, abgefragt werden, ob die Stringvariable A$ eine Zeichenkette enthält. Dazu muß das Programm um eine Zeile ergänzt



werden, die die erforderliche »IF«-Abfrage enthält. Enthält A$ nichts, also nur einen Leerstring, so ist der Stringvariablen eine beliebige Zeichenkette zuzuweisen.

```
19 IF A$ = "" THEN A$ = "ZEICHEN"
20 ...:LOAD A$,8
```

?NEXT WITHOUT FOR:

Erscheint, wenn zur Anweisung NEXT die entsprechende FOR-Anweisung fehlt. Dies kann verschiedene Gründe haben:

1) Im Programm befinden sich mehr NEXT- als FOR-Anweisungen.

2) Es wird versehentlich mit dem Basic-Befehl GOTO in eine FOR..NEXT-Schleife gesprungen.

3) Der Name der Schleifenvariablen (Variable hinter dem FOR-Befehl) in der FOR-Anweisung stimmt nicht mit derjenigen hinter der NEXT-Anweisung überein.

4) Es wurde innerhalb einer Schleifenkonstruktion mit dem Basic-Befehl GOSUB ein Unterprogramm aufgerufen, das eine NEXT- ohne entsprechende FOR-Anweisung enthält.

Fehler dieser Art lassen sich nur sehr schwer beheben. Vor allen Dingen dann, wenn das Programm lang und unübersichtlich geschrieben wurde. Der gesamte Programmablauf muß zu »Fuß« im Listing nachvollzogen werden. Zu jeder FOR-Anweisung ist die entsprechende NEXT-Anweisung zu suchen. Es muß überprüft werden, ob von irgendeiner Stelle in die Schleife gesprungen wird, in der der Fehler ausgegeben wurde.

Daher ist es sinnvoll, sich zu jedem Programm einen Ablaufplan (Flußdiagramm) anzufertigen, damit der Überblick nicht verloren geht und Fehler, die nur schwer zu beheben sind, vermieden werden.

?NOT INPUT FILE:

Erscheint, wenn versucht wird, mit den Befehlen INPUT # und GET # aus einer Datei etwas zu lesen, die zum Schreiben geöffnet wurde oder wenn als Filenummer (erste Zahl hinter dem OPEN-Befehl) »0« eingesetzt wird. Versuchen Sie mal im Direktmodus

```
OPEN0,3 <RETURN>
```

einzugeben. Der Computer meldet sofort einen »NOT INPUT FILE«-Error. Wenn eine Datasette zur Verfügung steht, erzeugt das folgende Beispiel die gleiche Fehlermeldung:

```
10 OPEN1,1,1,"TEST":REM BANDDATEI ZUM
SCHREIBEN ÖFFNEN
20 PRINT#1,"WORT":REM DAS WORT »WORT« IN DIE
DATEI SCHREIBEN
30 INPUT#1,A$:REM ERZEUGT DIE FEHLERMELDUNG
40 CLOSE1:REM KANAL SCHLIESSEN
```

Erscheint ein »NOT INPUT FILE«-Error, ist die hinter der Fehlermeldung angegebene Zeile zu LISTen. Ist dort die logische Filenummer hinter dem OPEN-Befehl »0«, so ist sie durch eine andere Zahl zu ersetzen. Vorsicht! Alle Ein-/Ausgabe-Befehle, die sich auf diesen OPEN-Befehl beziehen, müssen ebenfalls entsprechend geändert werden.

Steht in der geLISTeten Zeile kein OPEN-, sondern ein INPUT #- oder GET #-Befehl, dann ist versucht worden, aus einer Datei etwas zu lesen, die nur zum Schreiben geöffnet wurde (siehe Beispiel). Sollte dieser Fall eingetreten sein, ist zuerst zu überprüfen, ob der INPUT #- oder GET #-Befehl an der entsprechenden Stelle richtig ist. Wenn der Befehl stimmt, ist eine falsche Filenummer gewählt worden oder die fehlerhafte Zeile wurde mit dem GOTO- beziehungsweise GOSUB-Befehl angesprungen, ohne zuvor die geöffnete Datei zum Schreiben zu schließen.

?NOT OUTPUT FILE:

Es wurde versucht, mit dem Basic-Befehl PRINT # etwas in eine Datei zu schreiben, die nur zum Lesen geöffnet wurde. Auch dieser Fehler bezieht sich nur auf das Arbeiten mit Datasette. Beispiel:

```
10 OPEN 1,1,0,"TEST":REM KASSETTENDATEI ZUM LESEN
ÖFFNEN. ES MUSS EINE DATEI MIT DEM NAMEN TEST
EXISTIEREN.
20 A$="ABC":PRINT#1,A$:REM DIESE ZEILE ERZEUGT
DEN FEHLER
30 CLOSE1:REM KANAL SCHLIESSEN
```

Die gleiche Fehlermeldung erscheint auch dann, wenn man die Tastatur (Geräteadresse 0) mit dem Befehl »OPEN 1,0« zum Lesen öffnet und versucht, mit dem Befehl »PRINT # 1,"ABC"« etwas zum Kanal mit der Filenummer »1« zu schicken. Das heißt, die Tastatur läßt sich nicht als Ausgabegerät verwenden.

Tritt der Fehler in einem Programm auf, ist er genauso zu beseitigen wie der »NOT INPUT FILE«-Error.

?OUT OF DATA:

Die Fehlermeldung erscheint, wenn das Programm auf den Basic-Befehl READ stößt und entweder keine DATAs vorhanden sind oder schon alle gelesen wurden. Beispiel:

```
READ A$ <RETURN>
```

Wird der Befehl im Direktmodus eingegeben, erscheint der »OUT OF DATA«-Error, weil keine DATAs vorhanden sind.

Meldet der Computer diesen Fehler, sind entweder DATAs vergessen worden einzugeben oder die Schleifenvariable innerhalb einer FOR..NEXT-Schleife zählt zu weit. Beispiel:

```
10 FOR I=0 TO 10
20 READ A
30 NEXT I
40 END
50 DATA 1,2,3,4,5
```

Die Schleifenvariable »I« zählt von 0 bis 10, erwartet also 11 DATAs. Es sind aber nur fünf DATAs vorhanden. Folglich meldet der Computer einen »OUT OF DATA«-Error.

Wie der Fehler beseitigt wird, dürfte damit wohl auch geklärt sein. Hinweis: Der Fehler tritt zwar in der Zeile auf, deren Zeilennummer hinter der Fehlermeldung ausgegeben wird, befindet sich aber niemals in dieser Zeile, sondern immer in der Zeile, die die entsprechenden DATAs enthält.

?OUT OF MEMORY:

Dieser Fehler kann verschiedene Ursachen haben:

1) Der Speicherplatz im Computer reicht nicht für das Programm und die Variablen. Häufig tritt der Fehler auf, wenn DIM-Anweisungen zu groß gewählt wurden (zum Beispiel »DIM A(10000)«.

2) Wenn Maschinenprogramme geladen wurden, deren Startadresse über dem freien Basic-Bereich liegen (zum Beispiel SMON). In einem solchen Fall schafft der Befehl NEW Abhilfe, nachdem das Programm geladen wurde.

3) Verschachtelungstiefen bei FOR...NEXT-Schleifen

(maximal 10) oder Unterprogrammaufrufen, mit dem Befehl GOSUB (maximal 24) wurden überschritten. Die häufigste Fehlerursache entsteht dadurch, daß in einem Unterprogramm der Befehl RETURN fehlt.

Wurde die Verschachtelungstiefe überschritten, muß das Programm anders aufgebaut werden. FOR...NEXT-Schleifen lassen sich zum Beispiel durch IF-Abfragen ersetzen. Beispiel

```
10 FOR I=0 TO 10
20 A=A+1
30 NEXT I
```

ist identisch mit

```
10 A=A+1
20 I=I+1
30 IF I <>10 THEN 10
```

?OVERFLOW:

Entweder liegt das Ergebnis oder ein Zwischenergebnis eines Ausdrucks außerhalb des zulässigen Zahlenbereichs. Jedes Ergebnis muß zwischen »+/-1.37E38« liegen. Beispiel:

```
PRINT 9*2E37/100 <RETURN>
```

Wird dieses Beispiel eingegeben, erscheint der »OVERFLOW«-Error. Formt man das Beispiel um, so läßt sich dieser Ausdruck durchaus berechnen:

```
PRINT 9/100*2E37
```

Der Fehler ist dadurch entstanden, daß das Zwischenergebnis »9 * 2E37« den zulässigen Zahlenbereich überschritten hat. Im Falle eines »OVERFLOW«-Errors ist also ein Ausdruck so abzuändern, daß kein Zwischenergebnis den zulässigen Wert überschreitet.

?REDIM'D ARRAY:

Das Programm trifft zweimal auf die selbe DIM-Anweisung.

Eine Feldvariable (zum Beispiel A(15)) darf in einem Programm nur ein einziges Mal dimensioniert werden.

Beispiel:

```
10 DIM A(20)
20 DIM A(30)
```

Es tritt ein »?REDIM'D ARRAY ERROR IN 20« auf. Dieser Fehler läßt sich meistens dadurch vermeiden, daß alle benötigten DIM-Anweisungen in die erste Programmzeile verlegt werden. Weiterhin sollte darauf geachtet werden, daß auf alle möglichen DIM-Zeilen kein GOTO-Sprung erfolgt. Wenn sich dies nicht vermeiden läßt, kann man ja ein CLR: vor den DIM-Befehl setzen. Dann werden allerdings auch alle anderen Variablen gelöscht.

Beispiel:

```
10 CLR : DIM A(20)
20 GOTO 10
```

Diesmal tritt kein Fehler auf.

?REDO FROM START:

Eine INPUT-Anweisung erwartet die Eingabe einer Zahl; es wurden jedoch Buchstaben oder Zeichen eingegeben. Die ganze INPUT-Anweisung wird noch einmal bearbeitet.

Beispiel:

```
10 INPUT A
20 PRINT A
```

Starten Sie das Programmm mit »RUN« und geben Sie zum Beispiel ein »X« ein (plus »RETURN«-Taste). Der Computer wird dies mit der beschriebenen Fehlermeldung quittieren und die INPUT-Anweisung noch einmal ausführen. Es dürfen nur Zahlen eingegeben werden! Ausnahme ist der Buchstabe »E« zur Kennzeichnung von Exponential-Zahlen (zum Beispiel 1.5E+3, also 1,5 mal 10 hoch 3)

Tip: Überprüfen Sie Ihre Eingabe noch einmal sorgfältig auf unerlaubte Zeichen.

?RETURN WITHOUT GOSUB:

Der Computer trifft auf ein »RETURN«, ohne daß zuvor ein »GOSUB« aufgetreten ist.

Beispiel:

```
10 RETURN
```

Wenn Sie das Programm starten, tritt sofort der genannte Fehler auf. Aber: Es kann auch sein, daß zwar *eine* GOSUB-Anweisung auftritt, aber der Computer *einen zweiten* RETURN-Befehl findet. Häufigste Ursache: Ein Programm mit einem oder mehreren Unterprogrammen hat nach Beendigung des Hauptteils keine END-Anweisung und »läuft« deshalb unbeabsichtigt in die erste Unterroutine. An deren Ende trifft der C64 dann auf ein RETURN, ohne daß jedoch ein GOSUB gegeben wurde.

Beispiel:

```
10 REM HAUPTPROGRAMM
20 GOSUB 100
30 REM HAUPTPROGRAMMENDE
100 REM UNTERPROGRAMM
110 RETURN
```

Haben Sie den Fehler enddeckt? Zuerst wird in Zeile 20 die Unterroutine ab Zeile 100 korrekt mit GOSUB angesprungen und mit RETURN beendet. Dann jedoch bearbeitet der C64 Zeile 30, Zeile 100 und dann Zeile 110! Abhilfe: Am Ende des Hauptteils eine END-Anweisung. In obigem Beispiel:

```
40 END
```

?STRING TOO LONG ERROR:

a) Eine Stringvariable enthält mehr als die erlaubten 255 Zeichen, oder

b) Es wurde mittels INPUT# eine Zeichenkette eingelesen, die mehr als 89 (!) Zeichen enthält.

Beispiele:

a)

```
10 A$="X"
20 FOR I=1 TO 500
30 A$=A$+"X"
40 PRINT LEN(A$)
50 NEXT I
```


In Zeile 30 wird der String A$ immer um ein Zeichen erweitert. Zeile 40 informiert über die aktuelle Länge des Strings. Sobald er mehr als 255 Zeichen enthält, wird ein »?STRING TOO LONG ERROR IN 30« ausgegeben.

b)

```
10 OPEN 1,8,1,"TEST,S,W"
20 FOR I=1 TO 100
30 PRINT#1,"X";
40 NEXT I
50 CLOSE 1
60 OPEN 1,8,0,"TEST,S,R"
70 INPUT#1,A$
80 PRINT A$
90 CLOSE 1
```

Der erste Programmteil (Zeilen 10 bis 50) eröffnet eine Datei auf Diskette und schreibt 100 Zeichen direkt hintereinander in diese Datei. Direkt hintereinander deshalb, weil in Zeile 30 nach dem PRINT#-Befehl ein »;« steht.

Der zweite Programmteil (Zeilen 60 bis 80) versucht nun. Diese 100 Zeichen auf einmal in die Variable A$ einzulesen (was natürlich nicht funktioniert).

Um diesen Fehler besser zu verstehen, muß man sich die Funktionsweise des INPUT#-Befehls einmal vor Augen führen: INPUT# liest solange Zeichen in die angegebene Variable, bis in der Datei das erste »carriage return« (CHR$(13)) folgt. Dieses Zeichen, auch kurz »cr« genannt, wird beim Schreiben von Daten in eine Datei immer genau dann gesendet, wenn der PRINT#-Befehl nicht mit einem »,« oder »;« endet. In unserem Beispielprogramm wird in Zeile 30 durch das »;« nach dem PRINT#-Befehl das Senden des »cr« verhindert. Deshalb meldet der Computer beim INPUT# in Zeile 70 einen STRING TOO LONG ERROR, sobald er merkt, daß nach dem 89. Zeichen immer noch kein »cr« in der Datei aufgetaucht ist. Die Daten werden nämlich nicht sofort in die

Variable A$ eingelesen, sondern erst in einen Zwischenpuffer; und dieser kann nicht mehr als 89 Zeichen aufnehmen. Um unser kleines Programm doch noch lauffähig zu machen, müßte man den INPUT#-Befehl durch eine FOR-NEXT-Schleife mit einer GET#-Anweisung ersetzen (diese liest immer nur ein einziges Zeichen). Also:

```
70 FOR I=1 TO 100 : GET#1,B$ : A$=A$+B$ : NEXT I
```

Wenn in einem Programm ein STRING TOO LONG Error in einer Zeile mit einer INPUT#-Anweisung auftritt, ist der Fehler also nicht in dieser Zeile, sondern in der Routine zu suchen, die die Daten in die Datei geschrieben hat. Diese muß vor dem Senden eines Strings über die LEN-Funktion erst einmal testen, ob er die vorgeschriebene Länge von 89 Zeichen nicht überschreitet.

?SYNTAX ERROR:

Kann mehrere Gründe haben:

a) Generell ist der Syntax eines Befehls, also seine Schreibweise, von Ihnen falsch eingegeben worden.

b) In einer DATA-Zeile steht ein Zeichen oder eine Zeichenfolge, obwohl der READ-Befehl eine Zahl erwartet.

c) Das erste Zeichen des Basic-Speichers ist nicht Null.

Beispiele:

a) Allgemein tritt der Syntax-Error auf, wenn der Basic-Interpreter einen Befehl nicht versteht, also wenn Sie sich beim Schreiben eines Programms vertippt haben. Überprüfen Sie daher die Zeile, in der der Fehler auftrat, noch einmal sorgfältig, ob Sie die Befehle alle richtig geschrieben oder irgendein Zeichen vergessen haben! Häufigster Tippfehler aller Programmierer: »RUM« statt »RUN«.

Achtung! Dieser Fehler tritt auch auf, wenn in einer Variablenzuweisung die Anzahl der geöffneten Klammern ungleich Null ist. Beispiel: `A=(2+(3*4)`. Es fehlt eine »Klammer zu«. Oder: `A=2+(3*4))`. Eine Klammer wurde zu viel geschlossen.

b)

```
10 READ A,B,C
20 DATA 67, Z, 20
```

Es tritt ein Syntax-Error in Zeile 20 (!) auf, da die READ-Anweisung in Zeile 10 drei Zahlen als DATA-Werte erwartet. Die »zweite Zahl« ist jedoch ein Buchstabe.

Tip: Wenn Sie in DATA-Zeilen Zahlen und Buchstaben zusammen verwenden möchten und es tritt dieser Fehler auf, dann überprüfen Sie, ob Sie die Werte auch in der richtigen Reihenfolge auslesen.

c) Besonders tückisch: Wenn aus irgendeinem Grund der Inhalt der ersten Basic-Adresse nicht Null ist, so tritt bei der Eingabe von »RUN« oder »NEW« ein Syntax-Error auf. Dies ist meistens der Fall, wenn Sie den Basic-Start (Adressen 43 und 44) verändert haben, zum Beispiel, um eine Basic-Erweiterung zu verwenden.

Abhilfe: `POKE(PEEK(43)+PEEK(44)*256—1),0`

?TOO MANY FILES:

Es wurden mehr als die maximal erlaubten zehn Dateien (über einen OPEN-Befehl) eröffnet.

Beispiel:

```
10 OPEN 1,3
20 OPEN 2,3
30 OPEN 3,3
40 OPEN 4,3
50 OPEN 5,3
60 OPEN 6,3
70 OPEN 7,3
80 OPEN 8,3
90 OPEN 9,3
100 OPEN 10,3
110 OPEN 11,3
```

Sicherlich, dies ist ein extremes Beispiel, das im »Alltagsbetrieb« nie auftreten wird, aber ebenso ist es mit dieser Fehlermeldung. Sollte also bei Ihnen dieser Fehler einmal auftreten, so sollten Sie sich ernsthaft Gedanken darüber machen, ob Sie wirklich so viele offene Datenkanäle benötigen.

?TYPE MISMATCH:

Tritt auf, wenn versucht wird, eine Zeichenkette in einer numerischen Variablen abzulegen.

Beispiele:

```
A="TEST" oder A=A$
```

Untersuchen Sie die Zeile, in der der Fehler aufgetreten ist. Meistens wurde nur das $-Zeichen vergessen.

?UNDEF'D FUNCTION:

Eine mathematische Funktion wurde angesprochen, ohne vorher definiert worden zu sein.

Beispiel:

```
10 A=FN X(2)
```

Wenn Sie trotzdem der Meinung sind, die entsprechende Funktion definiert zu haben, so überprüfen Sie den verwendeten Funktionsnamen noch einmal. Zur Veranschaulichung hier nocheinmal der Syntax der FN-Anweisung:

Definieren: DEF FN name(variable)=funktion

Aufruf: variable=FN name(wert oder variable)

Ein korrektes Beispiel:

```
10 DEF FN TEST(X)=X*2
20 A=FN TEST(2)
30 PRINT A
```

?UNDEF'D STATEMENT:

a) Über RUN, GOTO oder GOSUB wurde eine Zeile angesprochen, die nicht existiert.

b) Es wurde versucht, eine Variable als Sprungziel zu verwenden.

Beispiel:

a)

```
10 GOTO 100
20 END
```

Dieser Fehler ist äußerst einfach zu beheben:

Zeile, in der der Fehler auftrat, listen, und überprüfen, ob die angesprochene Zeile wirklich vorhanden ist. Dann die GOTO- oder GOSUB-Anweisung entsprechend ändern.

b)

```
10 A=100 : GOTO A
100 END
```

Obwohl die Zeile 100 existiert, tritt der Fehler auf. Es darf nämlich nicht über eine Variable zu einer bestimmten Zeile gesprungen werden. Es gibt aber die Möglichkeit, über einen ON-GOTO-Befehl einen berechneten Sprung auszuführen:

```
10 INPUT "SPRUNG NACH ZEILE?";A
20 A=A/100 : ON A GOTO 100,200,300
100 PRINT "ZEILE 100" : END
200 PRINT "ZEILE 200" : END
300 PRINT "ZEILE 300" : END
```

Beachten Sie aber, daß der ON-GOTO-Befehl wie folgt arbeitet:

Wenn A=**1**, dann GOTO 100, wenn A=**2**, dann GOTO 200, ... Deshalb muß die eingegebene Zahl auch erst in dieses Format umgerechnet werden (`A=A/100`).

?VERIFY ERROR:

Durch ein VERIFY-Kommando wurde festgestellt, daß das Programm im Speicher mit dem auf Datasette beziehungsweise Diskette nicht übereinstimmt. Deutet bei Datasettenbetrieb meistens auf ein fehlerhaftes Band hin. Verwenden Sie dann eine neue Kassette.

Ist bei Diskettenbetrieb äußerst unwahrscheinlich, da die Floppy beim Speichern eines Programms automatisch einen Verify durchführt. Eventuell schadhafte Disketten werden daher schon frühzeitig erkannt. Generell kann man sagen, daß beim Arbeiten mit einer Floppy das »Verifizieren« entfallen kann. Nützlich dagegen ist es bei der Programmentwicklung, wenn man nicht sicher ist, ob man die aktuelle Version schon gespeichert hat. Hier bringt ein Verify die Antwort.

(ah/tr)



BÜCHER

C64-Computer-Handbuch

Auf der Titelseite heißt es: »Einführung und Referenz für kompetentes Arbeiten«. Und das ist es in der Tat. Das Buch ist eine deutsche Übersetzung und Bearbeitung der englischen Originalausgabe »Programming the Commodore 64« und kann getrost als eines der wenigen Standardwerke zum C64 gelten.

Auf über mehr als 600 Seiten findet jeder, der sich etwas intensiver mit dem C64 beschäftigen will, eine Menge an Informationen, Wissenswertes, Tips und Tricks, Grundlagen und Hinweise für Profis. Betrachtet man alleine die Kapitel über Basic, wird sogar Spezialisten in dieser Hinsicht das Lesen bestimmt nicht langweilig. Man merkt mit jeder Seite, daß dieses Buch von einem Könner mit langer praktischer Erfahrung geschrieben wurde, ohne überflüssigen Ballast, konzentriert und doch an wichtigen Stellen ausführlich genug. Das Buch ist in 17 Kapitel aufgeteilt, zusätzlich kommt ein Anhang mit wichtigen Tabellen sowie ein ausführliches Stichwortregister. Einen guten Eindruck von dem Umfang der behandelten Themen vermittelt ein Einblick in das Inhaltsverzeichnis:

Kapitel 1 und 2: Dem Vorwort folgt eine allgemeine Einführung über die Eigenschaften des C64.
3. C64/SX64-Basic zum Nachschlagen. Alle Befehle des C64 mit vielen interessanten Details.
4. Effektives Programmieren in Basic. Optimierung von Basic-Programmen mit vielen Beispielen.
5. Architektur des C64. Einführung in die Hardware des C64.
6. C64/SX64-Basic für Professionelle, wie Besonderheiten des Basic und eine Reihe von Dienstprogrammen und Erweiterungen.
7. Einführung in die Maschinensprache des 6510. Umfassende Beschreibung der CPU 6510 mit vielen Beispielen und Problemlösungen.
8. Typische Methoden der C64-Maschinenprogrammierung. Wie man das Kernel, Basic-Routinen und das RAM unter dem ROM nutzt sowie Ändern von Basic-Befehlen etc.
9. Verbindung von Basic und Maschinencode.
10. Der Befehlssatz der CPU 6510.
11. ROM-Führer. Speicherbelegung und Betriebssystem.
12. Grafik
13. Ton und Musik
14. Band
15. Diskette
16. Die Spiele-Ports. Joysticks, Drehregler, Grafiktablett, Maus und Lichtgriffel etc.
17. Drucker, Plotter, Modems.

Dieses Buch kann jedem wärmstens empfohlen werden, der sich etwas näher mit dem C64 beschäftigen will, sei es nur in Basic oder auch in Maschinensprache. Ein Handbuch, das garantiert nicht im Regal verstaubt. (gk)

Info: Raeto West, C64-Computer-Handbuch, TeWi-Verlag, 600 Seiten, ISBN 3-921803-24-1, Preis 66 Mark.

Basic-Grundkurs mit dem C64

Das Buch verspricht im Untertitel, »eine leichtverständliche Einweisung in Basic und die Datenkommunikation mit vielen Beispielen« zu geben. Und genau das ist auch der Fall. Auf 370 Seiten wird eine komplette Einführung in das Arbeiten mit dem C64 geboten, angefangen bei der Bedienung der Tastatur, Datasette und Floppy-Laufwerk über erste Schritte in die Welt der Basic-Programme bis hin zur Programmierung von Dateien und Datensammlungen.

Das Buch ist so konzipiert, daß auch ein absoluter Computer-Neuling schon sehr bald mit seinem C64 vertraut wird und selbständig kleinere Programme verfassen kann. Dabei werden wirklich keinerlei Voraussetzungen an den Leser gestellt. Wer beispielsweise als Anfänger nicht weiß, was eine Variable ist, wie man mit DATA und RESTORE umgeht oder wie man die Bildschirmausgabe formatieren kann, der wird gründlich darüber aufgeklärt und kommt schnell zu einem tieferen Verständnis der Arbeitsweise seines Computers. Auch für den Einsteiger oft etwas geheimnisvolle Basic-Spezialitäten wie die Statusvariable ST oder die Zeitvariable TI$ werden ausführlich erläutert.

Ein wichtiger Abschnitt des Buches beschäftigt sich mit der Verwaltung und Speicherung von Daten in DATA-Anweisungen oder einer sequentiellen Datei. Wie im gesamten Buch finden sich auch in diesem Kapitel viele kleinere Beispiel-Listings, die zum Ausprobieren und zu eigenen Erweiterungen einladen. Auch die Kommunikation zwischen zwei Computern über eine serielle Schnittstelle wird beschrieben.

Ein ausführlicher Anhang enthält eine kurze Beschreibung des gesamten Basic-Vokabulars, eine Liste aller Steuerzeichen und acht Seiten nützliche PEEKs und POKEs zum C64.

(Christian Quirin Spitzner/ev)

Info: Michael Hegenbarth, Raimund Trierscheid: Basic-Grundkurs mit dem C64, Markt & Technik, 370 Seiten, ISBN 3-89090-045-3, Preis 44 Mark.

50 Video-Spiele ausführlich erklärt

Erst der Untertitel »Anleitungen, Tips + Techniken für die bekanntesten C64-Spiele« zeigt, was das kleine Büchlein tatsächlich leisten will. Aktualität kann man bei dem aus dem Jahre 1984 stammenden Buch bei der rasanten Entwicklung selbstverständlich nicht mehr erwarten.

Von den damals auf dem Markt befindlichen Spielen für den C64 wurden die am meisten verbreiteten Spiele von den beiden Autoren herausgesucht und eingehend besprochen. Unter den ausgewählten Spielen sind unter anderem Soccer, Summer Games und David's Midnight Magic.

Man erfährt zunächst Hersteller, Programmstart und Anzahl der Mitspieler. Vermißt wird hier die nicht unwichtige Angabe des empfohlenen Verkaufspreises. Anschließend wird in Stichworten die Steuerung des Spielablaufes gegeben. Jedes Spiel wird sachlich nach Grafik, Sound und Beeinflußbarkeit des Spielverlaufes beurteilt. In der ausführlichen Beschreibung des Spieles liegt vielleicht der größte Nutzen gegenüber den häufig nur auf Englisch vorliegenden Anleitungen. Zusätzlich dienen viele Tips zum Spielverlauf dazu, optimale Ergebnisse zu erzielen. Mit Kritik wird nicht gespart. Der unter »Unsere Meinung« abgedruckten kritischen Gesamtbewertung kann man sich voll anschließen.

Unter den oben dargestellten Einschränkungen gibt das Buch zuverlässige Entscheidungshilfen und gute Anleitungen.

(D. Hein/ev)

Info: M. Frey, H. J. Rothe: 50 Video-Spiele ausführlich erklärt, Luther-Verlag, 109 Seiten, ISBN 3-88707-048-8, Preis 29,80 Mark.

Computerlösungen für Schule und Studium (Band 2)

Wie der Titel schon andeutet, wendet sich das vorliegende Buch mehr an die mathematisch Interessierten als an »Otto Normalverbraucher«. Die behandelten Themen sind ausschließlich Stoff der Oberstufe am Gymnasium und decken die gesamte Palette bis zum Abitur ab.

Aufbauend auf den ersten Band werden zunächst verbesserte Verfahren zur Differential- und Integralrechnung entwickelt. Dann geht es mitten in die Ausgleichs- und Näherungsrechnung, was auch alle angehenden und professionellen Wissenschaftler interessieren wird, die jeden Tag ellenlange Meßreihen auszuwerten haben. Auch die Kapitel über Experimente zur Wahrscheinlichkeit und Statistik sowie über Stochastik eignen sich durchaus für ernsthafte Anwendungen. Es wird ein umfangreiches Stochastik-Programm entwickelt, mit dem von der Berechnung der Fakultät bis zur Ausgabe der empirischen Verteilung alles möglich ist.

Aus den weiteren Themen wie geometrische Muster und Kegelschnitte, Kugelgeometrie und Körperdrehungen im Raum sowie der analytischen Geometrie, möchte ich noch die dreidimensionalen Funktionsdarstellungen herausgreifen. Wieder findet man am Schluß einer interessanten Herleitung und Entwicklung ein interessantes Programm, mit dem 3D-Funktionen auf dem Bildschirm dargestellt werden können.

Die Autoren wollen das Buch nicht als Hausaufgabenerleichterung oder Mogelhilfe verstanden wissen. Vielmehr ist der Band ein »mathematisches Arbeitsbuch«. Sicherlich können hier nicht alle Weisheiten der Mathematik vermittelt werden, doch zur Auffrischung oder Erweiterung bereits vorhandenen Wissens eignet es sich hervorragend. Der Leser wird nicht gleich mit dem nackten Listing konfrontiert. In jedem Kapitel wird zuerst das Problem betrachtet und analysiert und dann das Programm ausführlich hergeleitet. Dabei dienen viele Abbildungen zur Verdeutlichung des Sachverhalts.

Alle Programme sind in Basic, teilweise auch in Simons Basic, geschrieben. Das Umschreiben auf andere Basic-Erweiterungen wird aufgrund der ausführlichen Herleitungen nicht schwerfallen.

Das Buch ist ein Leckerbissen für Schüler der gymnasialen Oberstufe, Studenten mit mathematischer Ausrichtung und natürlich Hobbymathematiker. Sie erhalten ein Werk, in dem klar und verständlich höhere Mathematik auf dem Computer umgesetzt wird.

(Thomas Tai/ev)

Info: Rainer und Patrick Gaitzsch: Computerlösungen für Schule und Studium Band 2, Moderne Verlagsgesellschaft, 78 Seiten, ISBN 3-478-09280-9, 29,80 Mark.

Das Kassettenbuch

Dieses Buch beschreibt in aller Ausführlichkeit (als Ergänzung zum Commodore-Handbuch) das Zusammenarbeiten zwischen Computer und dem Datenrecorder. Da ist zunächst die Beschreibung aller Befehle, die mit dem Zugriff des VC 20 oder C64 auf die auf Band gespeicherten Daten und Programme zu tun haben. Dies bezieht sich sowohl auf Basic, als auch auf die entsprechenden Kernelroutinen, die das Arbeiten mit Maschinenprogrammen ermöglichen. Natürlich fehlt es auch nicht an guten Ratschlägen, die sich auf den Recorder selbst beziehen, wie zum Beispiel die Pflege der Andruckrollen oder das Nachjustieren des Schreib-/Lesekopfes. Auch eine Mithörkontrolle und ein Selbstbau-Kassetteninterface wird, wenn auch etwas kurz, beschrieben. Der zweite Teil des Buches bezieht sich auf ein Fasttape-Programm, mit dem es möglich ist, Programme und auch Daten 10- bis 20mal schneller abzuspeichern. Für den VC 20 (und dies betrifft viele Käufer dieses Buches) ist leider eine Programmversion abgedruckt, die nur bei einem voll ausgebauten (28 KByte) Speicher funktioniert. Um die Routinen in einem anderen Speicherbereich laufen zu lassen, müssen sie vom Leser (!) umgeschrieben werden. Auf diese Routinen aufbauend, werden dann einige andere Programme wie beispielsweise ein Kassettenverzeichnis oder ein Backup-Programm für Disketten- oder Kassettenprogramme entwickelt. Diese Routinen sind alle ausführlich beschrieben und kommentiert. Was den Profi jedoch enttäuschen wird, ist das fast gänzliche Fehlen einer Beschreibung der im Betriebssystem verankerten Kassettenroutinen. Auch eine in diesem Zusammenhang notwendige Beschreibung der Ein-/Ausgabebausteine fehlt in diesem Buch. Dafür findet man einen Hinweis auf andere Data Becker-Bücher – eine sicherlich nicht sehr befriedigende Situation. Auch bezüglich des Aufzeichnungsformates sind die Ausführungen des Autors sehr knapp gehalten und obendrein scheinbar direkt den »Tips & Tricks«-Handbüchern des gleichen Verlages entnommen.

Wegen der gut lesbaren und informativen Einführung in das Arbeiten mit der Datasette ist das Kassettenbuch mit seinen zahlreichen Programmen trotz einiger Schwächen eine rentable und empfehlenswerte Anschaffung für den fortgeschrittenen Einsteiger.

(Christoph Sauer/ev)

Dirk Paulissen: Das Kassettenbuch zu C64 und VC 20, Data Becker, 192 Seiten, ISBN 3-89011-030-4, 29 Mark.

Das Schulbuch zum Commodore 64

Schüler, die den Computer für etwas Wichtigeres als Spielen benutzen wollen, werden vielleicht von diesem Schulbuch zum Commodore angesprochen. Das Buch soll den Computer zum Lehrer machen. Neben reinen Trainingsprogrammen, wie beispielsweise Vokabellernen, gibt es auch Programme, die bestimmte Probleme mit Hilfe des Computers lösen sollen. So können der größte gemeinsame Teiler und das kleinste gemeinsame Vielfache von zwei Zahlen bestimmt oder Vokabeln alphabetisch geordnet werden. In Umfang und Schwierigkeit decken sich die im vorliegenden Schulbuch dargestellten Aufgaben etwa mit dem Stoff der Klassen 5 bis 10. In mehr als 40 kurzen Programmen werden ausgewählte Probleme aus Mathematik, Physik, Chemie, Biologie/Ökologie, Fremdsprachen, Geschichte, Erdkunde und Wirtschaftskunde behandelt.

Die Vorgehensweise erfolgt stets nach dem gleichen Schema:

1. Vorstellung des Problems
2. Problemanalyse
3. Flußdiagramm
4. Programmlisting
5. Variablenliste
6. Programmbeschreibung
7. Ergebnisausgabe

Alle sieben Schritte werden in einfacher, verständlicher Sprache sehr ausführlich dargestellt. Ein interessierter Leser dürfte bei der Arbeit kaum Schwierigkeiten haben. Um einfache Programme zu erhalten, wurde auf Schönheit und komfortables Arbeiten bewußt verzichtet. Von Basic werden nur die einfachsten Grundelemente verwendet.

Das Lernen erfolgt vorwiegend bei der gründlichen Stoffanalyse und der computergerechten Aufbereitung. Den größten Lernerfolg wird der Schüler haben, der die im Buch gegebenen Anregungen zum Ausbau der Programme aufgreift. Bei genügender Selbstdisziplin kann der Computerfreak mit dem Buch mehr lernen als durch teure Nachhilfestunden.

Für Tippfaule und Leute, die nur mit den fertigen Programmen arbeiten wollen, hält Data Becker eine Programmdiskette bereit, die auch noch einige zusätzliche Programme enthält. Das umfangreiche Begleitheft zur Programmdiskette gibt alle für die Arbeit nötigen Informationen.

(D. Hein/ev)

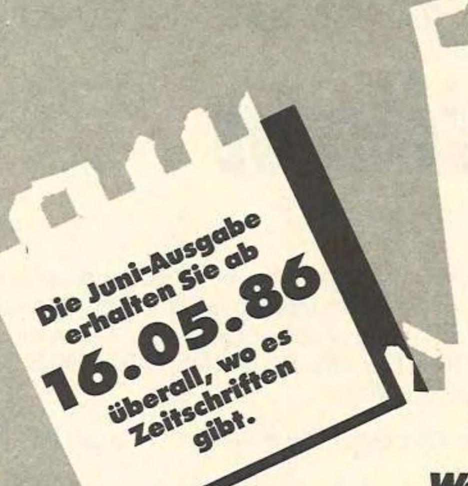
Info: Prof. Dr. W. Voß: Das Schulbuch zum Commodore 64, Data Becker, 330 Seiten, ISBN 3-89011-019-3, Preis 49 Mark.

Das Data-Becker-Lexikon zum Commodore 64

Dieses Buch darf jedem Einsteiger empfohlen werden, der sich von den vielen Fachausdrücken in der Computerliteratur etwas verunsichert fühlt und einen schnellen, allgemeinen Überblick bekommen möchte. Eine Vielzahl von Begriffen zu Computern im allgemeinen und zum Commodore 64 im besonderen wird in verständlicher und zumeist hinreichend ausführlicher Form besprochen. Vor einem Mißverständnis muß allerdings gewarnt werden: Dieses Lexikon kann ein Lehrbuch und die intensive Beschäftigung mit dem Computer genausowenig ersetzen, wie man durch Pauken von Vokabeln ein flüssiges Englisch lernen kann. Als Nachschlagewerk für den einen oder anderen Fachbegriff und als Orientierungshilfe für den Einsteiger erfüllt es jedoch seinen Zweck recht gut.

(ev)

Jordan/Schellenberger: Das Data-Becker-Lexikon zum Commodore C 64, Data Becker, 353 Seiten, ISBN 3-89011-013-4, 49 Mark.



Viele Wege führen zu 64'er, dem Magazin für Computer-Fans:

Wer programmiert, gewinnt!

Gewinnen Sie mit Ihren Software-Ideen in den monatlichen Leserwettbewerben von »64'er« Geld und tolle Preise!

Besonders Computer-INTERESSIERTE, NEUGIERIGE, EINSTEIGER,

die sich mit den Commodore-Computern C16, C64 oder C128 beschäftigen, zeigt »64'er« den schnellsten Weg zum erfolgreichen Umgang mit Ihrem Computer durch ausführliche Grundlagenbeiträge, Kurse und jede Menge Tips & Tricks.

Als Fortgeschrittener begleitet Sie »64'er« dann mit aktuellen Beiträgen, Tests, interessanten Anwendungs-Listings und praktischen Hardware-Basteleien.

In der Juni-Ausgabe lesen Sie:

■ Großer Messebericht von der CeBIT '86 in Hannover: »64'er« zeigt die neuesten Trends auf dem Computer-Markt. ■ Zubehör für Commodore-Computer: »64'er« stellt nützliche Extras für Monitore, Drucker, Floppies und Computer vor. ■ Datenbanken: Marktübersichten über Dateiverwaltungen. ■ Digitalisierung von Sprache ■ Listing des Monats: Super Textverarbeitung (Text +) ■ Weitere interessante Listings: Disketten-Labels automatisch gedruckt. — Kalender selbst drucken ■ Für Anfänger und Profis: Jede Menge Tips & Tricks für C16, C64 und C128. ■ Auswertung eines »64'er«-Musik-Wettbewerbes: Alle Gewinner werden aufgeführt und das Sieger-Listing »Shades« vorgestellt.

Falls Sie »64'er« noch nicht regelmäßig beziehen, sichern Sie sich jetzt Ihr persönliches Abonnement und nutzen die damit verbundenen Vorteile: ■ Sie beziehen »64'er« ohne Mehrkosten bequem per Post frei Haus ■ Sie haben Ihr »64'er« bereits bei sich zu Hause — noch bevor Sie es bei Ihrem Zeitschriftenhändler kaufen können. ■ Sie sind sicher, keine Ausgabe zu versäumen.

Sie erhalten — wenn Sie zur Anforderung den nebenstehenden Gutschein verwenden — auf alle Fälle die neueste Ausgabe als Probeheft unverbindlich und kostenlos.

Gutschein

FÜR EIN KOSTENLOSES PROBEEXEMPLAR DES 64'er-MAGAZINS

JA, ich möchte »64'er«, das Magazin für Computerfans, kennenlernen.
Senden Sie mir bitte die aktuellste Ausgabe kostenlos als Probeexemplar. Wenn mir »64'er« gefällt und ich es regelmäßig weiterbeziehen möchte, brauche ich nichts zu tun: Ich erhalte »64'er« dann regelmäßig frei Haus per Post und bezahle pro Jahr DM 78,— (Ausland auf Anfrage).

Vorname, Name

Straße　　　　PLZ, Ort

Datum　　　　1. Unterschrift

Mir ist bekannt, daß ich diese Bestellung innerhalb von 8 Tagen bei der Bestelladresse widerrufen kann und bestätige dies durch meine zweite Unterschrift. Zur Wahrung der Frist genügt die rechtzeitige Absendung des Widerrufs.

Datum　　　　2. Unterschrift

Gutschein ausfüllen, ausschneiden, in ein Kuvert stecken und absenden an: Markt&Technik Verlag Aktiengesellschaft, Vertrieb, Postfach 1304, 8013 Haar

64S586

Computer-Lexikon

Oft gesucht und nie gefunden – eine übersichtliche Erklärung der wichtigsten Begriffe aus der Computertechnik.

Jeder Einsteiger hört immer wieder von den verschiedensten Begriffen aus der Computertechnik, doch weiß er teilweise nicht, was sie im einzelnen bedeuten. Dieses kleine Lexikon soll ihm helfen, sich etwas besser in dieser Welt der Fremdworte zurechtzufinden.

Akkumulator (Accumulator)
Ein spezielles > Register des Prozessors, in dem die meisten logischen Rechenoperationen durchgeführt werden.

Akustikkoppler (Acoustic coupler)
Ein Zusatzgerät, das Datensignale in akustische Signale umwandelt, so daß diese über ein Telefon an einen anderen Computer übertragen und von diesem empfangen werden können. Natürlich muß an der Empfangsstation auch ein Akustikkoppler oder > Modem vorhanden sein. Dies ermöglicht den Datenaustausch mit anderen privaten und öffentlichen Informationssystemen. Dabei beträgt die übliche Datenübertragungsgeschwindigkeit 300 > bit/s, das entspricht etwa 30 Zeichen pro Sekunde. Der Unterschied zum Modem besteht darin, daß er nicht direkt ans Telefonnetz angeschlossen werden kann; dadurch werden Sicherheit und Übertragungsgeschwindigkeit geringer. Allerdings braucht ein Akustikkoppler im Gegensatz zum Modem nicht eigens vom Benutzer bei der Post angemeldet zu werden. Eine FTZ-Nummer genügt.

Algorithmus (Algorithm)
Es gibt verschiedene Arten, ein Problem oder eine Aufgabe zu lösen. Ein Algorithmus bezeichnet nur eine solche Methode, ein Problem »anzupacken«.

Adresse (Address)
Die Adresse gibt an, wo sich die Daten im Speicher befinden (ähnlich Hausnummern in einer langen Straße). Die Speicherplätze sind durchnumeriert. Dadurch werden ihnen Adressen zugeteilt, über die dann auf den Inhalt der Speicherzelle zugegriffen werden kann. Der Inhalt kann gelesen und/oder überschrieben werden.

Alphanumerische Zeichen (Alphanumeric)
Alle Zeichen, die ein Computer über eine Eingabeeinheit, zum Beispiel die Tastatur, akzeptiert und mittels einer Ausgabeeinheit darstellen kann; mindestens aber das Alphabet und die Dezimalzahlen.

ALU (Arithmetic Logic Unit)
»Arithmetisch-logische Einheit«, der Teil des > Mikroprozessors, der sämtliche Rechenoperationen durchführt.

Anweisung (Statement)
Ein einzelner oder ein zusammengesetzter Ausdruck (Zahlen, Variablen, Operationen etc.) innerhalb eines Programms, der dem Computer mitteilt, was er ausführen soll.

Anwendungsprogramm (Applications program)
Ein »Dienstleistungsprogramm« mit meist speziellem Anwendungbereich (zum Beispiel Buchhaltung, Textverarbeitung), im Gegensatz etwa zu einem Systemprogramm, mit dessen Hilfe zum Beispiel auch Anwendungsprogramme verschiedenster Art erstellt werden können.

Array
Ein (mehrdimensionales) Datenfeld, auf dessen einzelne Elemente zugegriffen werden kann. Ein Schachbrett ist zum Beispiel ein Array mit acht mal acht Elementen. Jedes einzelne Kästchen stellt ein Element dar, und jedes einzelne Element hat einen eigenen Wert.

ASCII (American Standard Code for Information Interchange)
Ein Standardcode, der einer Menge von 128 Zeichen (Zahlen, Buchstaben, Symbolen) ein aus 7 > Bit bestehendes Muster zuweist. Dieser Standard soll die Kommunikation zwischen verschiedenen Computern und Systemen erleichtern. Ein kleines »s« zum Beispiel wird repräsentiert durch »1110011«.

Assembler
1. Bezeichnet eine maschinenorientierte Programmiersprache, die von Prozessor zu Prozessor unterschiedlich ist; Assemblerprogramme können nur zusammen mit »ihrem« Prozessortyp laufen, im Gegensatz zu den höheren Programmiersprachen, die unabhängig von dem eingesetzten Mikroprozessortyp sind. Der Vorteil besteht darin, daß Assemblerprogramme die Fähigkeit »ihres« Mikroprozessortyps am optimalsten ausnutzen und – erst einmal in Maschinencode übersetzt – am wenigsten Speicherplatz beanspruchen, was eine kürzere Programmdurchlaufzeit zur Folge hat. Häufig werden Assemblerprogramme bei zeitkritischen Prozessen auch in ein Programm, das in einer höheren Programmiersprache wie zum Beispiel Basic geschrieben ist, eingebunden.
2. Übersetzungsprogramm, das einen Assembler-Quellcode in den binären, ablauffähigen Maschinencode umwandelt. Dabei werden die Assembler-Befehle – auch Mnemonics oder Operationscode genannt – auf Formulierungsfehler (Syntax-Fehler) überprüft und in Maschinensprache übersetzt.

Austesten (Debugging)
Fehlersuche in einem Programm, idealerweise durch einen Probelauf mit Testdaten, der alle möglichen Situationen durchtestet, die innerhalb eines Programms auftreten können.

Basic (Beginner's All-purpose Symbolic Instruction Code)
Einfache Programmiersprache, die ein Programmieren im > Dialogmodus ermöglicht. Sie ist leicht erlernbar und hat dadurch im Heimcomputer-Bereich eine weite Verbreitung gefunden.

Baud
Maßeinheit (veraltet) für die Übertragungsgeschwindigkeit von Daten. Im allgemeinen Fall die Übertragunsrate in > Bits pro Sekunde. Als Faustregel gilt für die > serielle Byteübertragung eine tatsächliche Übertragungsrate von Baudrate/10, da zur Übertragung eines Zeichens 8 Bit + 1 Startbit + 1 Stoppbit üblich sind.
Die Einheit Baud wurde vor kurzem abgelöst durch bit/s.

BCD (Binary Coded Decimal notation)
> Binäre Darstellung dezimaler Ziffern mit 4 > Bits. Dabei bleiben 6 Bitmuster unbenutzt, da mit 4 Bits 16 Zahlen darstellbar sind. Viele > CPUs haben spezielle Befehle zur Manipulation von BCDs.

Befehl (Instruction)
Kommandoanweisungen, die den Computer bestimmte Operationen ausführen lassen. Auch Zuweisungen, wie zum Beispiel »X=5+6« (in > Basic), sind Befehle, wobei die rechte Seite »5+6« ein Ausdruck ist, dessen Wert der Variablen »X« zugewiesen wird. (Es existieren auch Programmiersprachen, die gänzlich ohne Befehle auskommen, sogenannte »funktionale Sprachen«, die nur Ausdrücke kennen, wie zum Beispiel (reines) > LISP.

Betriebssystem (Operation system)
Ein Programm, das die »Fähigkeiten« eines Computers so organisiert, daß > Anwendungsprogramme lauffähig und damit benutzbar werden. Es wirkt wie eine Art Puffer, der ein und dieselbe > Software an die > Hardware verschiedener Computer anpaßt und dafür verwendbar macht.

Bildschirmeditor (Screen Editor)
Ein Textbearbeitungsprogramm, das es ermöglicht, mit Hilfe von > Cursorbewegungen jede beliebige Stelle eines Textes, der in Arbeit ist, auf dem Bildschirm anzusteuern und zu korrigieren, zu ersetzen, zu streichen und so weiter.

Binär (Binary)
Ein Zahlensystem mit der Basis zwei, bei dem die Zahlen nur durch Kombinationen der Ziffern 0 und 1 dargestellt werden. Da es sich leicht in einem Entweder-Oder-Schema (positive oder negative Spannung, Anwesenheit oder Abwesenheit von Spannung, schwarze oder weiße Punkte auf dem Bildschirm und so weiter) repräsentieren läßt, wurde es zum »Charakteristikum« digitaler Computer.

Bit (Binary dig)
Kleinste mögliche informationstragende Einheit, im > binären Zahlensystem, symbolisiert durch eine 1 oder eine 0. Darstellbar in Alternativen wie: ja oder nein, an oder aus, positiv oder negativ.

Bus
Datenübertragende Verbindungsbahnen innerhalb eines Computers oder zwischen einem Computer und seinen Peripheriegeräten. Ein Bus kann auch für spätere, vorläufig nicht aktivierte Gebrauchsweisen »offen« sein. Ein Bus ist in der Regel mit einer größeren Anzahl von Datenursprungs- und Zieladressen verbunden.

Byte
Informationseinheit aus einer bestimmten Anzahl zusammengruppierter > Bits (in der Regel 8, ein Byte aus 8 Bits kann jeden Wert von 0 bis 255 enthalten). Ein Wort ist aus zwei, ein Langwort (long word) aus vier Byte zusammengesetzt; ein > Nibble umfaßt ein halbes Byte (4 Bits). Ein Byte repräsentiert oft die kleinste adressierbare Einheit im > Speicher.

Centronics-Schnittstelle
Schnittstelle, die von der Firma Centronics zum Anschluß ihrer Drucker an Mikrocomputer verwendet wird. Es handelt sich um eine Schnittstelle, die die zu druckenden Zeichen seriell, aber bitparallel verarbeitet, das heißt, alle Bits eines Zeichens werden mit Hilfe von acht parallelen Leitungen übertragen. Die Centronics-Schnittstelle wurde mittlerweile zu einem quasi-Standard für die parallele Zeichenübertragung im Zusammenhang mit Druckern.

Character
Bedeutet übersetzt Zeichen.

Chip
Ein Chip ist ein rechteckiges Siliziumplättchen, das nach einigen Ätz- und Diffusionsvorgängen bestimmte elektrische Funktionen erfüllt. Diese Chips findet man im Computer in den käferartig geformten ICs, wo sie die verschiedensten Funktionen erfüllen können (Mikroprozessoren, Soundgeneratoren, Speicher- oder Logikbausteine etc.).

Code
1) Allgemein: eine Vorschrift für Darstellungsweise von Informationen.
2) Darstellung einer Information in symbolischer, leicht »prozessierbarer« Form.
3) Die Ausdrücke und > Befehle, aus denen ein Programm zusammengesetzt ist.

Compiler
Ein komplexes Programm, das Hochsprachen (wie > Pascal oder > Fortran) in > Maschinensprache übersetzt, das heißt in das »Befehlsformat« des Mikroprozessors umwandelt. Für die Entwicklung des ersten Fortran-Compilers wurde ein Aufwand von zwölf »Mannjahren« berechnet.

Computer der fünften Generation
(Fifth generation computers)
Computergeneration im Entwicklungsstand (vor allem in Japan und Amerika), die sehr hohe > Hochsprachen (wie > Prolog) als > Maschinensprache besitzen. Sie sollen der Erforschung der sogenannten > Künstlichen Intelligenz (KI) neue Bereiche eröffnen.

Computergenerationen (Computer generations)
Bezeichnet die aufeinanderfolgenden Entwicklungsgruppen von (Digital-)Computern, von denen sich jede durch eine technische Gemeinsamkeit auszeichnet. Die 1. Generation durch Relais, die 2. durch Röhren, die 3. durch Transistoren, die 4. durch Integrierte Schaltungen, die 5. durch VLSI (Very Large Scale Integration).

CP/M
Ein weitverbreitetes > Betriebssystem von Digital Research für 8-Bit-Computer mit > Diskettenverwaltung.

CPU (Central Processing Unit)
Die zentrale Verarbeitungs- und Steuereinheit eines > Computers. Verwaltet sämtliche Betriebsmittel (> Peripherieeinheiten und > Speicher) des Computers.

Cursor
Bewegliches Lichtsignal auf dem Bildschirm, das die Position des jeweils nächsten Ausgabezeichens markiert.

Datei (> File)
Dateneinheiten (auch Programme), die auf speziellen magnetischen Speichermedien (früher Lochkarten, heute Kassetten, > Disketten, Magnetbänder oder Magnetspeicherplatten, aber auch im Computer-Hauptspeicher) gespeichert und abrufbar gehalten werden können.

Dialog
Mit Computern kann man auf zwei Arten verkehren: im Stapel-Verfahren (der Benutzer gibt die für ein Programm nötigen Daten auf einmal hintereinander ein und erhält dann nach einiger Zeit das Ergebnis), auch Batch-Betrieb genannt, oder im Dialog-Verfahren. Beim Dialog-Verfahren reagiert das System sofort anschließend an die Frage oder die einzelne Dateneingabe des Benutzers, oder anders ausgedrückt: Es findet ein ständiger Wechsel zwischen Benutzereingabe und Computerausgabe statt; der Dialog wird dabei durch ein Programm gesteuert. Beispielsweise soll eine Abrechnung erstellt werden: Der Computer fragt nacheinander alle Daten ab, und der Benutzer antwortet über Tastatureingabe auf die jeweilige Frage mit der Eingabe einer Information, solange, bis die betreffende Rechnung erstellt ist.

Digital
Darstellung von Daten durch Auflösung in diskrete Signale (zum Beispiel 0 Volt/5 Volt). Im Gegensatz zur »analogen«

Darstellung, die ihre Grenze in der Präzision der verwendeten physikalischen Geräte findet, ist die digitale Datendarstellung prinzipiell beliebig verfeinerbar.

Diskette (> Floppy disk)
Magnetisch beschichtete runde Folie in Papier- oder Plastikhülle, die als mittelschneller (Massen-)Datenspeicher Verwendung findet (vom Prinzip her genauso wie ein Magnetband).

DOS (Disk Operating System)
Teil eines > Betriebssystems, das die > Dateien auf den > Floppy-Disks verwaltet, zum Beispiel einer Datei Speicherraum auf einer Diskette zuweist und auch die Funktionen des Diskettenlaufwerks gewährleistet.

Editieren (Editing)
Veränderung oder Modifikation einer Textdatei durch Einfügungen, Verschiebungen, Löschungen und so weiter.

Editor
Ein Programm, das das > Editieren von Textdateien ermöglicht.

Eingabe (Input)
Alles, was über »Eingabekanäle«, zum Beispiel die Tastatur, dem Computer als Information zugeführt wird. Die Eingabekanäle sind > Schnittstellen des Computers zu seiner Umgebung, quasi die Wahrnehmungskanäle eines Computers. Seine Sinnesorgane wären entsprechend die Tastatur, die > Maus, das Disketten- oder Kassettenlaufwerk, der > Akustikkoppler.

EPROM (steht für: Erasable programmable read-only memory)
Halbleiterspeicher, der vom Computer nur gelesen, aber nicht beschrieben werden kann. Auf EPROMs kann man zum Beispiel eigene Programme »brennen« (in das EPROM dauerhaft schreiben). Ein EPROM kann mit UV-Licht gelöscht und mit einem EPROM-Programmiergerät neu beschrieben werden.

Festkommazahl (Fixed-point number)
Zahlendarstellung mit fixierter Kommastelle, die im kommerziellen Bereich benutzt wird.

File
Bedeutet übersetzt Datei. Ein File ist eine bestimmte Datenmenge, die auf einem Datenträger gespeichert ist und die einen eindeutigen Namen besitzt, über die sie angesprochen werden kann.

Floppy
Floppy ist eine Kurzform für > Floppy-Disk, aber auch das Diskettenlaufwerk wird häufig als Floppy bezeichnet.

Floppy-Disk
Die Floppy-Disk, häufig auch als > Diskette oder nur als > Floppy bezeichnet, ist zur Zeit das wichtigste Massenspeichermedium für kleine und mittlere Personal Computer. Sie besteht aus einer flexiblen Kunststoffscheibe mit magnetischer Oberfläche. Diese Scheibe rotiert ständig in einer Hülle. Nur die für die Funktion wichtigen Bereiche sind von außen zugänglich; dies sind das Mittelloch, das Spuranfangsloch (Indexloch) und der Schlitz für den Schreib/Lesekopf. Disketten sind in vier Größen auf dem Markt: 3-, 3½, 5¼- und 8-Zoll-Durchmesser.

Der Trend, auch im Hinblick auf höhere > Speicherkapazitäten, geht immer mehr zur 5¼-Zoll-Diskette. Die 3½-Zoll-Floppy kam erst vor kurzem auf den Markt und gewinnt langsam an Bedeutung. Die typischen Speicherkapazitäten betragen 0,5 bis 2 MByte (8 Zoll), 0,1 bis 2 MByte (5¼-Zoll) und 250 bis 750 KByte (3½ Zoll). Obwohl beide Seiten der rotierenden Datenträger mit einem magnetischen Material beschichtet sind, wird häufig nur eine benutzt, entsprechend ist die Schutzhülle auch nur auf einer Seite mit dem Schreib-/Leseschlitz versehen. Disketten sind zur Zeit die beliebtesten Massenspeicher für Personal Computer, zumal sie neben hoher Speicherkapazität bei geringen Kosten (4 bis 20 Mark pro Diskette) auch über einen Direkt-Zugriff über die Angabe von Spur und Sektor verfügen – im Gegensatz zu Speichermedien, die nur einen sequentiellen Datenzugriff ermöglichen (Kassette).

Flußdiagramm (Flowchart)
Schematische Darstellung eines sequentiellen Programmablaufs unter Verwendung von standardisierten Zeichen und Symbolen, die Ereignisse oder Beziehungen zwischen Ereignissen bezeichnen können.

Forth
Programmiersprache auf der Basis eines »gefädelten« (threaded) > Codes, sehr schnell, aber mit unleserlichem Sprachbild. Wird von manchen Programmierern eher für eine »Religion« als eine Programmiersprache gehalten.

Gleitkommazahl (Floating-point number)
Zahlendarstellung, bei der sich die Kommastelle der Zahlengenauigkeit beziehungsweise dem Zahlenwert anpaßt. Für den wissenschaftlichen Bereich und den Geschäftsbereich geeignet.

Hardcopy
Ausdruck von Texten oder Grafiken, die am Bildschirm zu sehen sind, über einen Drucker auf Papier.

Hardware
Alle mechanischen oder elektronischen Teile eines Computersystems, die man »anfassen« kann – aber nicht unbedingt anfassen sollte.

Hexadezimal
Das hexdezimale Zahlensystem beschreibt die Darstellung von Zahlen zur Basis 16. Die Dezimalzahlen 0 bis 15 werden durch die Ziffern 0 bis 9 und die Buchstaben A bis F dargestellt. Das Hexadezimalsystem wird zur übersichtlicheren Darstellung von Dual- oder Binärzahlen (Basis 2) benutzt. Die Binärzahl »1011« entspricht dezimal »11« und hexadezimal »B«; dieses hexadezimale »B« wird zur besseren Unterscheidung von dem Buchstaben »B« mit »B « oder als »hex B« beziehungsweise »hexadezimal B« bezeichnet. Hexadezimale Zahlen werden bevorzugt in der Mikrocomputertechnik eingesetzt, da sich ein Byte mit nur zwei Symbolen eindeutig beschreiben läßt; diese Darstellung ist für den Programmierer wesentlich einfacher verständlich als der Binärcode (Beispiel »FB« ist übersichtlicher als »11111011«, insbesondere, wenn es sich um größere Ausdrucke handelt). Die meisten Computer machen einen Speicherauszug (Dump) in der hexadezimalen Darstellungsart. Bei Einplatinen-Computern sind die Hex-Tastaturen (Tastaturen, die nur über die hexadezimalen Ziffern 0 bis 9 und A bis F verfügen) weit verbreitet.

Hexadezimal Darstellung (hexdecimal notation)
Darstellung von Zahlen auf der Basis 16. Als Zeichen werden die Zahlen 0 bis 9 und die Buchstaben A bis F benutzt. Hexadezimalzahlen werden in der Regel durch ein vorangestelltes »$« oder »&« bezeichnet. »$B« ist eine Zahl im Hex-Code, die binär als »11111011« dargestellt werden kann.

Höhere Programmiersprache (Higher level language), auch Hochsprache genannt.
Alle Programmiersprachen, die sich durch echte Worte programmieren lassen (im Gegensatz zu > Assembler), und bei denen die Anweisungen nicht im Befehlsformat des > Mikroprozessors formuliert sind, sondern bei denen sich ein Befehl in ganzen Gruppen für den Mikroprozessor direkt ver-

stehbarer Befehlsfolgen niederschlägt. Sie sind für den Menschen leichter lesbar und erhöhen die Programmiergeschwindigkeit, aber auf Kosten der Ausführungsgeschwindigkeit und des Speicherplatzes. Der Computer kann Hochsprachen über einen > Compiler oder einen > Interpreter verarbeiten. Ein Compiler übersetzt ein Programm aus der Hochsprache in eine für die > CPU direkt verarbeitbare Befehls- und Datensequenz; der Interpreter reagiert auf jeden Befehl einer Hochsprache, indem er eine für diesen Befehl in ihm abgelegte Befehlssequenz in > Maschinensprache (Maschinencode) durchläuft. Ein compiliertes Programm ist in der Ausführung schneller als ein Interpreter, erfordert aber einen höheren Erstellungsaufwand als ein Programm, das von einem Interpreter »übersetzt« wird. Der Vorteil eines Interpreters ist, daß man mit seiner Hilfe Programme im > Dialogmodus erstellen kann. Gewöhnlich wird zum Beispiel > Pascal compiliert, > Basic aber interpretiert.

Integrierte Schaltung (Integrated circuit)
Eine Ansammlung sehr kleiner elektronischer Schaltungseinheiten auf einem > Chip.

Interaktiv (Interactive)
Der Computer reagiert sofort auf Eingaben des Benutzers, ein »Dialog« entsteht; im Gegensatz dazu gibt der Benutzer im sogenannten > batch processing ein Programm erst vollständig ein, »schickt es nach unten« an den Computer, der erst reagiert, nachdem er das vollständige Programm verarbeitet hat. Mit dem C64 arbeitet man interaktiv. Batch processing gibt es in der Regel nur bei großen Computern.

Interpreter
Durchläuft für jeden Befehl einer bestimmten Programmiersprache eine festgelgte Befehlssequenz in > Maschinensprache. Diesen Prozeß nennt man »interpretieren«. Programme, die mittels eines Interpreters erstellt wurden, setzen zu ihrer Ausführung wiederum einen Interpreter voraus. Basic ist zum Beispiel eine Interpretersprache. Man spricht deshalb auch von einem Basic-Interpreter.

Iteration
Wiederholung von Schleifen (Durchläufen) eines Programmteils.

KByte
Man sagt zwar Kilo-Byte, jedoch hat 1 KByte nicht 1000, sondern 1024 Byte ($1024 = 2^{10}$). KByte ist die Maßeinheit für die Größe eines Speichers. Der C64 besitzt 64 KByte > RAM, das sind 65536 Byte (64 x 1024).

Künstliche Intelligenz
Ein »intelligenter« Computer soll sich mit dem Benutzer unterhalten können, Sprache und Bilder verstehen, Probleme selbständig lösen und wissen, wie es auf der Welt so zugeht. Die Wissenschaft, die Computern dies alles beibringen will, nennt sich Künstliche Intelligenz (KI). Die zentrale Frage, um die es in der KI geht, ist folgende: »Wie kann das, was Menschen wissen, im Computer dargestellt und verarbeitet werden?« Die Probleme der Künstlichen Intelligenz können mit den bisherigen Programmiermethoden nicht mehr gelöst werden. Man braucht geeignete Methoden, um die Dinge der realen Welt (beispielsweise Personen, Gegenstände, Gesetzmäßigkeiten, Zusammenhänge) auf dem Computer darzustellen. Der Computer soll ja die Realität kennenlernen, denn nur wenn er über die Welt, in der die Menschen leben, Bescheid weiß, kann er »intelligent« reagieren. Ein solcher Computer »weiß« zum Beispiel: »Bäume sind Pflanzen«.

Laderoutine (Booting or Bootstrapping)
Kleines Ladeprogramm, zumeist im > ROM, das Programme oder > Betriebssysteme – bei einer bestimmten Speicheradresse beginnend – lädt. Beim C64 gibt es so etwas nicht. Hier muß das Ladeprogramm erst in den Speicher geladen werden.

Laufwerk (Floppy-Laufwerk, Disketten-Laufwerk)
Ein > Peripheriegerät, das Informationen (> Bits) auf die magnetisierbare Oberfläche von > Disketten oder > Platten (besser: Festplatten) schreibt oder davon abliest.

Leistungstest (Benchmark)
Eine Anzahl von Programmen zum Ermitteln von Geschwindigkeit, Leistungsfähigkeit und Genauigkeit verschiedener Computer.

Logo
Ein einfacher Lisp-Abkömmling, graphikorientierte Programmiersprache für Mikrocomputer. Geeignet für den Programmierunterricht von Kindern.

Maschinencode (Maschine Code)
Eine dem Mikroprozessor unmittelbar verständliche Programmiersprache (Maschinensprache), in der alle Anweisungen als Bitmuster dargestellt sind.

Matrix (Matrix)
Punktmuster eines Zeichens auf dem Bildschirm oder dem Druckkopf eines Matrix- oder Nadeldruckers. In der Mathematik ein rechteckiges Zahlenschema, »Feld«, zur Beschreibung von Strukturen.

Maus (Mouse)
Ein auf dem Tisch frei bewegbares, auf einer Kugel gleitendes handgroßes Eingabegerät mit einer oder mehreren Tasten. Benutzerfreundlicher als Cursortasten, die zu ersetzen die wesentlichste Funktion einer Maus ist.

Mensch-Maschinen-Schnittstelle (Man-machine interface)
Alle »Berührungspunkte« zwischen Computer und Anwender, die eine Kommunikation zwischen beiden herstellen: Tastatur (> Maus), Bildschirm, Sprachein- und Sprachausgabe und so weiter.

Menü (Menu)
Auswahlliste der Teilfunktionen eines > Anwendungsprogramms.

Mikroprozessor (Microprocessor)
Ursprünglich eine integrierte Schaltung auf einem > Chip, die logische und arithmetische Operationen ausführen kann. Wird aber auch als Synonym für Mikrocomputer gebraucht.

Modem (MOdulator/DEModulator)
Ein Gerät, das den Anschluß eines Computers an datenübertragende Medien (zum Beispiel Telefon) ermöglicht. Vergleiche > Akustikkoppler.

Monitor
1. Der Bildschirm eines Computerterminals. Beim C64 kann nicht nur ein Monitor angeschlossen werden, sondern auch ein Fernsehgerät. Ein Monitor liefert jedoch ein besseres Bild.
2. Ein Programm, das die Kontrolle über den > Computer oder ein > Betriebssystem möglich macht. Einen Monitor braucht man erst dann, wenn man sich schon besser mit dem Computer auskennt und Erfarhungen mit Maschinensprache (Assembler) besitzt.

Netz (Network)
Netzartige Verbindung zwischen mehreren Computern, verdrahtet oder mittels Datenübertragungsgeräten wie > Modem oder > Akustikkoppler zum Zweck des Daten- und Informationsaustausches.

Niedere Programmiersprache (Low level language)
Programmiersprachen (zum Beispiel > Assembler), bei denen jede > Anweisung (im Idealfall) jeweils einer Anweisung im > Maschinencode entspricht. Oder umgekehrt eine

Sprache, in der es zu jedem direkt ausführbaren Maschinenbefehl ein künstliches Sprachkonstrukt gibt. Mit wachsender Computertechnik verlieren niedere Programmiersprachen immer mehr an Bedeutung.

Oktal (Octal)
Zahlensystem mit der Basis 8. Im Gegensatz zum Binär-, Hexadezimal- oder Dezimalsystem spielt das Oktalsystem in der modernen Computertechnik kaum mehr eine Rolle.

Offline/Online
Online: Bezeichnet die bestehende Verbindung zwischen dem Computer und einem > Peripheriegerät (zum Beispiel ein angeschlossener Drucker im Ready-Modus). Offline: wenn ein Gerät nicht vom Computer angesprochen werden kann (zum Beispiel ein ausgeschalteter Drucker oder eine vom Programm verlangte Diskette, die nicht im Laufwerk steckt).

Operator
1. Verknüpfung; zum Beispiel eine Verknüpfung zweier Elemente aus irgendeinem Bereich (etwa Zahlen), wobei das Ergebnis wieder im gleichen Bereich liegen muß. Zum Beispiel eine arithmetische Operation zwischen Zahlen, wobei das Ergebnis wiederum eine Zahl ergibt.
2. Bedienungs-/Überwachungspersonal bei Großrechnern, Mailboxen, Datenbanken und ähnlichem.

Parallele Schnittstelle (Parallel interface)
Im Gegensatz zu einer > seriellen Schnittstelle, bei der die > Bits nacheinander (seriell) übertragen werden, wird bei einer parallelen Schnittstelle ein Bitmuster, das ein Byte ausmacht, gleichzeitig übertragen. Eine parallele Datenübertragung ist daher in der Regel schneller als eine serielle.

Pascal

Eine compilierte höhere Programmiersprache, die von N. Wirth als Alternative zu Algol 68 entwickelt wurde. Eine strukturierte, schnelle und schnell zu compilierende Sprache. Geeignet zum Erlernen von Programmiertechniken.

Peripheres Gerät (Peripheral)
Alle »äußeren«, an einen Computer anschließbaren Geräte, wie Tastaturen, Drucker, > Modems, > Laufwerke und so weiter.

Pixel
Kleinster Punkt auf dem Bildschirm oder einem Matrixdrucker, wird auch »pel« genannt.

Platte (Disk)
Starre, magnetisch beschichtete Scheibe, die in ein Festplatten-Laufwerk eingebaut ist und normalerweise nicht ausgewechselt werden kann. Die Datenübertragungsgeschwindigkeit ist etwa um die Größenordnung 10 höher, die Speicherkapazität etwa um die gleiche Größenordnung größer als bei einem Disketten-Laufwerk.

Plotter
Ausgabegerät zum Erstellen von Zeichnungen mittels eines Schreibstiftes anstelle eines Druckkopfes (wie bei einem Drucker). Ein Plotter wird dort benötigt, wo sehr genaue und akkurate Zeichnungen notwendig sind (Architektur, Landvermessung, Meßtechnik, Konstruktion).

Port
> Schnittstelle zur Außenwelt aus der Sicht der > CPU, oft ein Adressenbereich im > Speicher, über den Eingabe und Ausgabe erfolgt. Beim C64 gibt es fünf Ports: Expansion-, User-, Kassetten-, Joystick- und den seriellen Port.

Portabilität (Portability)
Meist durch > Betriebssysteme ermöglichte Fähigkeit, ein und dieselbe Software auf verschiedenen Computern laufen zu lassen.

Programmentwicklung (Software engineering)
Systematisierte Programmentwicklung im Gegensatz zur »try and error«-Methode.

Prolog
Prolog wurde etwa 1970 in Marseille entwickelt. Ähnlich wie Lisp, die wohl bekannteste Sprache der > Künstlichen Intelligenz, unterscheidet Prolog sich grundlegend von Sprachen wie > Basic und > Pascal. Prolog ist ebenso wie > Basic eine interaktive Sprache. Die Entwicklung und Ausführung von Prolog-Programmen erfolgt im Dialog mit dem > Computer. Das ist aber auch schon die einzige Gemeinsamkeit von > Basic und Prolog. Denn diese Sprache beruht auf einem radikal neuen Konzept.

Der Programmierer braucht sich nicht mehr um > Algorithmen zur Lösung seines Problems zu kümmern, sondern muß genau angeben, worin sein Problem besteht. In herkömmlichen Programmiersprachen, wie auch zum Beispiel in > Basic, bestimmt der > Programmentwickler die Reihenfolge der Computeroperationen. Er legt sie nämlich mit den Programmbefehlen fest. In Prolog-Programmen wird nicht mehr das »wie« spezifiziert, sondern das »was«. Prolog wird vor allem dort eingesetzt, wo Symbole verarbeitet werden. Für numerische Datenverarbeitung, also Berechnungen und die Verarbeitung von Zahlen, ist diese Sprache nicht entworfen worden.

Typische Anwendungen von Prolog sind: Der Aufbau von Wissensbasen für Expertensysteme oder intelligente Datenbanksysteme und Verarbeitung natürlicher Sprache; sie umfaßt das Erkennen natürlicher Sprache und die Gesprächsführung durch das Programm Bilderkennung und -verarbeitung (Szenenanalyse).

Prozedur (Routine)
Teil eines Programms, das eine bestimmte, möglichst kleine Funktion durchführt. Prozeduren werden mit einem Namen aufgerufen, und Parameter können übergeben werden. Im Gegensatz zu > Pascal beispielsweise sind beim > Basic des C64 keine Prozeduren möglich.

Puffer (Buffer)
Zeitweiliger > Speicher, mit dessen Hilfe zum Beispiel verschiedene Datenübertragungsraten der > CPU und > Peripherie-Geräten ausgeglichen, »kompensiert«, werden. Man unterscheidet zwischen permanent in einem > Speicher reservierten Pufferzonen (beim C64 zum Beispiel der Kassettenpuffer) von vorübergehend als Puffer verwendeten Speicherbereichen.

RAM (Random Access Memory)
Halbleiterspeicher, der sowohl gelesen als auch beschrieben werden kann. Wenn der > Computer ausgeschaltet wird, geht der Inhalt dieses > Speichers verloren. Das ist nicht der Fall beim > ROM.

Register
Speichereinheit im > Mikroprozessor (der > CPU), in der Daten kurzzeitig während arithmetischer, logischer oder Übertragungsoperationen zwischengespeichert werden. Der 6510 im C64 besitzt zwei Register, das X- und das Y-Register. Zusätzlich existiert noch der > Akkumulator, eine besondere Art Register, mit dem alle Rechenfunktionen durchgeführt werden.

Rekursion
Eine bestimmte Problemlösungsmethode, bei der ein Problem in ein einfacheres Problem gleicher Art aufgelöst wird, bis der einfachste Fall gefunden ist. Aus dem wieder – in

Gegenrichtung gleichsam – der komplexe Fall aufgebaut wird. Auf diese Weise kann ein Problem scheinbar »durch sich selbst« gelöst werden.
Beispiel: Gegeben ist Fakultät (2)=2; gesucht ist Fakultät (X)=3; also: Fakultät (X); die rekursive Lösung lautet: Fakultät (Fakultät (X-1) * X).
Die Rekursion ist eine elegante Programmiermethode, die allerdings viel Speicherplatz benötigt: sie läßt sich oft auch durch > Iteration ersetzen.

Reserviertes Wort (Reserved word)
Ein Wort, das zum Sprachumfang einer Hochsprache gehört, auf diese Weise »reserviert« ist und daher nicht als Variablenname benutzt werden darf. Beim C 64 sind das alle Basic-Befehle und einige besondere Variable, zum Beispiel TI und ST.

ROM (Read Only Memory)
Halbleiterspeicher, der einmal beschrieben, weder gelöscht noch überschrieben werden kann (Nur-Lese-Speicher). Auch wenn keine Spannung anliegt, bleibt der Inhalt eines ROMs erhalten. Hingegen kann ein > EPROM gelöscht und wieder programmiert werden (Eraseabel Programmable ROM = löschbarer und programmierbarer Nur-Lese-Speicher).

RS232C
Standard für eine > serielle Schnittstelle. Manche Drucker werden über diese Schnittstelle betrieben, hauptsächlich wird sie jedoch zur Datenübertragung per Telefon eingesetzt. Der C 64 besitzt eine solche > Schnittstelle am User-Port, sie entspricht jedoch nicht der Norm, so daß zusätzlicher Aufwand notwendig ist, damit ein Gerät mit dieser Schnittstelle angeschlossen werden kann.

Schleife (Loop)
Progammteil, der so oft wiederholt wird, bis eine bestimmte Bedingung erfüllt ist. Im C 64-Basic gibt es lediglich die > Befehle FOR...NEXT und IF...THEN, mit denen eine Programm-Schleife möglich ist.

Schnittstelle (Interface)
Gemeinsame Grenze, der »Berührungspunkt« zwischen > Computer und anzuschließenden Geräten. Schnittstellen schaffen die Verbindung nach »draußen«, wandeln Daten so um, daß das Empfangsgerät (zum Beispiel ein Drucker) etwas mit diesen Daten anfangen kann.

Serielle Schnittstelle (Serial interface)
> Schnittstelle, an der, im Gegensatz zur > parallelen Schnittstelle, Daten sequentiell übertragen werden. Siehe auch > Baud. Verbreitete Norm ist die > RS232C-Schnittstelle.

Simulation
Modellhafte Nachahmung von (Lebens-)Situationen mit Hilfe eines Computers. Simulatoren werden dort eingesetzt, wo ein Versuch mit einem echten Gerät zu teuer werden würde, oder eine genaue Beobachtung des Versuchsablaufs nicht anders erreicht werden kann. Simulatoren werden in der Industrie eingesetzt (Lufthansa, Mercedes, BMW, etc.) und beim C 64 hauptsächlich als Spiel (Flight Simulator, Revs, Wirtschaftssimulator).

Software
Alle Programme, die von der > Hardware ausgeführt werden. Sie als C 64-Besitzer lernen Software in Form von Programmen auf > Diskette, Kassette oder in Zeitschriften als Listing kennen.

Speicher (Memory)
Der Computerbereich, in dem Daten in adressierbarer Form zwischengelagert werden können. Der C 64 besitzt 64 > KByte > RAM und 20 KByte > ROM.

Stapel (Stack)
Speicherbereich, in dem Daten für die Kontrolle von > Unterprogrammen (Subroutinen) abgelegt sind. Als Basic-Programmierer werden Sie ihn nur dann bemerken, wenn der C 64 einen »OUT OF MEMORY ERROR« meldet, obwohl der Hauptspeicher noch genügend Platz frei hat.

String
Die englische Bezeichnung für eine > Zeichenkette. Beim C 64 steht ein String immer zwischen Anführungsstrichen.

Strukturierte Programmierung (Structured programming)
Systematisierte Programmierung mit wenigen Kontrollstrukturen und ohne Sprünge (GOTOs). Eine echte strukturierte Programmierung ist mit dem > Basic des C 64 nicht möglich, kann jedoch mit etwas Aufwand simuliert werden. Basic-Erweiterungen oder andere Sprachen (> Pascal, Comal) stellen Befehle zur Verfügung, mit denen man strukturiert programmieren kann (REPEAT..UNTIL, DO..WHILE, IF..THEN..ELSE).

Terminal
1. > Peripheres Gerät, bestehend aus einer Tastatur und einem Bildschirm oder einem anderen Ausgabesystem.
2. Wenn der C 64 über das Telefon mit einem anderen > Computer verbunden ist, wird der Austausch der Daten mit einem Terminalprogramm geregelt.

Umgekehrte Polnische Notation (Reverse Polish Notation – RPN oder UPN)
Eindeutige Darstellung arithmetischer Ausdrücke ohne Klammern, wobei die > Operatoren hinter den ihnen zugeordneten Werten stehen. Die Computersprache > Forth funktioniert nach dem Prinzip der UPN, aber auch einige Taschenrechner, zum Beispiel die von Hewlett Packard. Sie erkennen UPN-Taschenrechner daran, daß keine =-Taste vorhanden ist, sondern eine ENTER-Taste. Mit diesen Computern läßt sich wesentlich schneller und einfacher rechnen als mit den üblichen. Allerdings ist das System etwas gewöhnungsbedürftig.

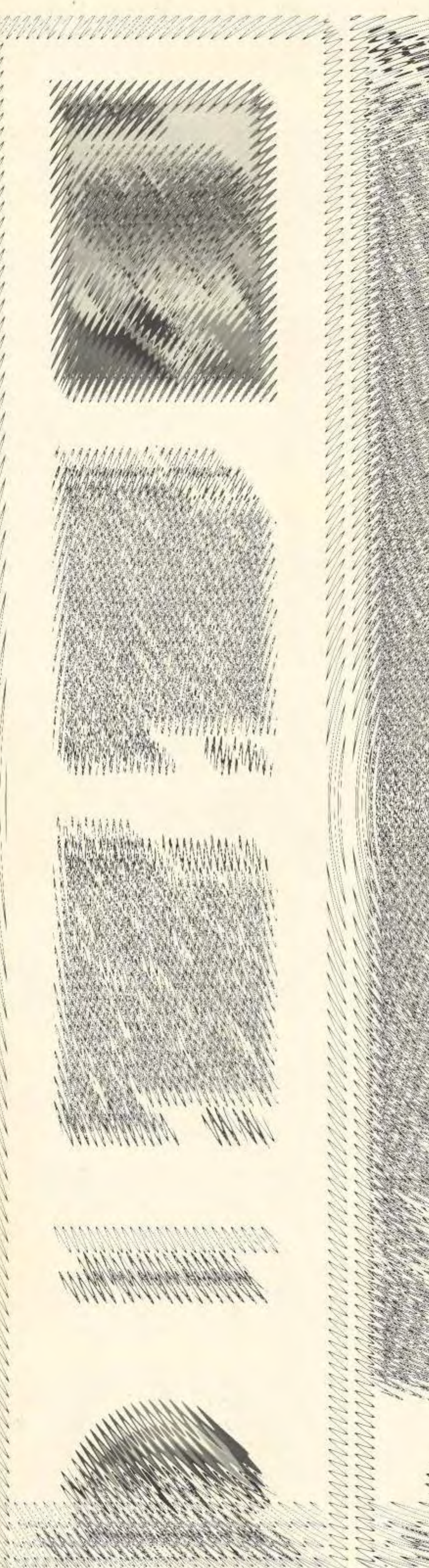
Unterprogramm (Subroutine)
> Prozeduren, die mehrmals während eines Programmlaufs aufgerufen werden. Beim C 64 werden Unterprogramme mit GOSUB aufgerufen und mit RETURN verlassen. Mit Unterprogrammen kann ein Programm sehr übersichtlich gestaltet werden.

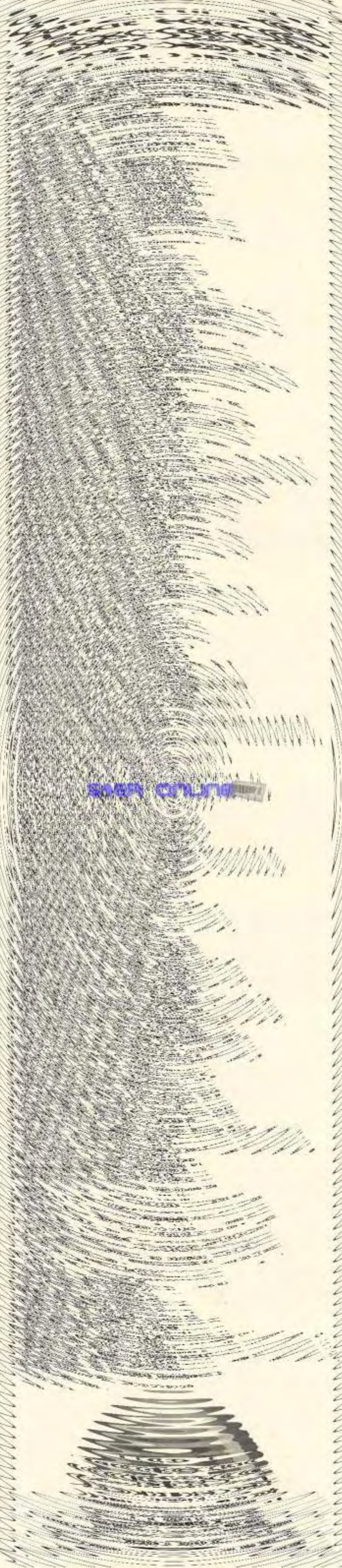
Variable
Name, an den ein Wert gebunden ist. Indem man sich (in einem Programm) auf den Namen bezieht, bezieht man sich auf den Wert, der an diesen Namen gebunden ist. Der C 64 kennt drei verschiedene Variablenarten:
1. Stringvariable (alle Zeichen, Ziffern und Buchstaben), gekennzeichnet durch ein $, Beispiel: A$= "STRING"
2. Realvariable (alle Zahlen), zum Beispiel A=-3.234
3. Integervariable (alle ganzen Zahlen (3,-3), gekennzeichnet durch das %-Zeichen, Beispiel: A%=123.

Zeichenkette (Character string)
Eine Folge von Zeichen zwischen Gänsefüßchen. In einem > String kann jedes Zeichen stehen, wie Buchstaben, Ziffern und andere Zeichen.

Zufallszahl (Random number)
Vom Computer generierte Zahl, die idealerweise weder wiederholbar noch voraussagbar ist. Da ein > Computer deterministisch arbeitet, kann er ohne speziellen > Hardware-Zusatz keine wirklichen Zufallszahlen, sondern nur Pseudo-Zufallszahlen (nach bestimmten > Algorithmen) erzeugen. Der Befehl, um beim C 64 eine Zufallszahl zu erzeugen, lautet RND(x).

(Peter Pörtner/cg)





Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael Scharfenberger
Stellv. Chefredakteur: Albert Absmeier
Koordination: Georg Klinge
Redaktion: Christine Geißler, Boris Schneider, Achim Hübner, Gerd Donaubauer, Arnd Wängler, Gottfried Knechtel, Karsten Schramm, Volker Everts, Thomas Röder

Titelfoto: Jens Jancke

Titelgestaltung: Heinz Rauner Grafik-Design

Layout:
Leo Eder (Ltg.), Sigrid Kowalewski (Cheflayouterin), Rolf Raß, Katja Milles

Herstellung: Klaus Buck

Auslandsrepräsentation:
Schweiz: Markt & Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042-41 56 56, Telex: 862 329
USA: M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94 063 Telefon: (4 15) 366-36 00

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten werden, so muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt & Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt & Technik Verlag AG verlegten Publikationen und dazu, daß Markt & Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Marketingleiter Vertrieb: Hans Hörl (114)
Vertriebsleitung: Helmut Grünfeldt (189)

Anzeigenverwaltung und Disposition: Michaela Hörl

Verlagsleiter M&T-Buchverlag: Günther Frank

Druck: SOV St.-Otto-Verlag GmbH, Laubanger 23, 8600 Bamberg

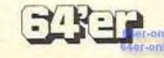
Preis: Das Einzelheft kostet DM 14,–

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1, Telefon (07 11) 6 48 30

Verantwortlich:
Für redaktionellen Teil: Michael Scharfenberger
Für Anzeigen: Britta Fiebig
Redaktions-Direktor: Michael M. Pauly

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 46 13-0, Telex 5-22 052






PROTEXT
TOP
ASS